VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro získávání statistik citačního ohlasu publikací skupin autorů

A System for Retrieving Statistics of Citations for Publications of Groups of Authors

2017

Bc. Jiří Littner

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jiří Littner**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Systém pro získávání statistik citačního ohlasu publikací skupin autorů

A System for Retrieving Statistics of Citations for Publications of Groups of Authors

Jazyk vypracování: čeština

Zásady pro vypracování:

Jedním ze základních hodnocení vědeckého výkonu autorů a institucí je citační ohlas publikací. Aktuálně existuje celá řada organizací spravující údaje o vědeckých publikacích ve vlastních informačních systémech resp. databázích publikací (např. Web of Science, Scopus, Google). Tyto informační systémy poskytují často jen omezenou funkcionalitu, typicky neposkytují statistiky citací pro skupiny autorů, katedry, případně fakulty. Cílem této práce je návrh a implementace systému umožňujícího snadno a přehledně zobrazit informace, které nejsou v těchto informačních systémech běžně k dispozici.

1. Vyberte alespoň dvě databáze publikací a nastudujte funkcionalitu informačních systémů těchto databází, zaměřte se na funkcionalitu týkající se citačního ohlasu publikací a možnosti identifikace autorů v těchto databázích.
2. Nastudujte aplikační rozhraní těchto databází a vytvořte infrastrukturu pro využití těchto rozhraní.
3. Navrhněte a implementuje systém umožnující získání informací o citačním ohlasu publikací, zaměřte se především na statistiky citačního ohlasu pro skupiny autorů, katedry a fakulty.
4. Výslednou aplikaci otestujte a vyhodnoťte.

Seznam doporučené odborné literatury:

Podel pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Michal Krátký, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Rád bych na tomto místě poděkoval doc. Ing. Michalu Krátkému, Ph.D. za rady poskytnuté při vypracování této diplomové práce, které mi pomohly práci zhotovit v konečné podobě a také bych rád poděkoval Mgr. Daniele Tkačíkové za potvrzení povolení Scopus API.

Abstrakt

Existuje několik organizací, které uchovávají data o vědeckých publikacích, jejich autorech a citacích, ale informační systémy těchto organizací neumožňují zobrazit statistiky pro skupiny autorů. Mým úkolem bylo navrhnout a vyvinout systém, který získá informace o citacích autorů naší univerzity z databází organizací Scopus a Web of Science. Výsledný systém měl data zpracovat, uložit a přehledně zobrazit statistiky citačního ohlasu jednotlivých autorů v informačním systému dle předem definovaných skupin včetně souhrnných statistik pro celé skupiny odpovídající rozdělení univerzity jako jsou fakulty nebo katedry. Proto byl vytvořen informační systém s přístupovými objekty, které přistupují k API citačních databází. Informační systém následně získané informace ukládá do vlastní databáze a přepočítává H-indexy autorů. Systém zobrazuje výsledné informace v uživatelském rozhraní a umožňuje základní úpravné operace nad skupinami, autory i jejich pracovními zařazeními. Skupiny je možné vytvořit a upravit v hierarchii odpovídající členění univerzity. Informace o citačním ohlasu jsou získávány a zpracovávány v pravidelných intervalech.

Klíčová slova: Citace, Publikace, Scopus, Web of Science, ASP.NET, JQuery, C#, Databáze

Abstract

Some organizations manage data of scientific articles their authors and citations, but the information systems of these organizations do not show statistics for the groups of authors. My task was to design and develop a system which obtains citations information from databases of organizations Scopus and Web of Science. Only citations of authors of our university should be obtained. The resulting system should process the data, stores them and display citations statistics of individual authors clearly in information system according to predefined groups. System should display summary statistics for the whole groups too. Groups are corresponding to division of the university such as faculties or departments. So an information system with access objects was created that accesses the API of citation databases. Information system then subsequently stores data in its own database and calculates H-Indexes of authors. System displays the resulting information in the user interface and also allows basic editing operations with groups, authors and their job titles. Groups can be created and edited in the hierarchy corresponding to the subdivision of the university. Information of citations are collected and processed at regular intervals.

Key Words: Citations, Publications, Scopus, Web of Science, ASP.NET, JQuery, C#, Database

Obsah

Seznam použitých zkratek a symbolů

SQL	–	Structured query language
CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
TFS	–	Team Foundation Server
URL	–	Uniform Resource Locator
ORM	–	Object-Relational Mapping
XML	–	Extensible Markup Language
SGML	–	Standard Generalized Markup Language
HTTP	–	Hypertext Transfer Protocol
REST	–	Representational State Transfer
DOM	–	Document Object Model
API	–	Application Programming Interface
LINQ	–	Language-Integrated Query
ISO	–	International Organization for Standardization
IS	–	Informační systém

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Pro zpracování této diplomové práce jsem si vybral téma blízké mému zaměření, a to jsou Databázové a informační systémy.

Na univerzitách po celém světě pracují vědečtí pracovníci, kteří píší odborné texty do vědeckých časopisů, konferenčních sborníků nebo knih. Při psaní jakéhokoliv typu odborného textu je obvykle nezbytné uvést odkazy na články, knihy nebo jiné dokumenty, ze kterých autor čerpal. Zde se nabízí dvě možnosti a to, že autor může ve své práci přímo citovat a použít tak pasáž původního dokumentu v souladu s autorským právem, nebo může využít myšlenky autora původního dokumentu jako základ pro vlastní argumentaci. Autor tak uvede svou práci do souvislosti s dosavadním stavem poznání, potvrdí myšlenky, kterými dospěl k vlastním závěrům nebo tak doloží fakta, o které se ve své práci opíral.

Publikované časopisecké články i jejich bibliografické odkazy se využívají jako zdroj informací v citačních a publikačních databázích, o nichž uvádím více v kapitole 2. Citace samotné jsou upraveny normou pro zpracování bibliografických citací ČSN ISO 690.

Já jsem se zaměřil právě na problematiku citací odborných publikací. Existuje několik organizací spravující informace o vědeckých publikacích, jejich autorech a citacích. Informační systémy těchto organizací však neumožňují rozdělení autorů do skupin odpovídajícím hierarchii univerzity, jako jsou fakulty, katedry nebo odborné skupiny. Proto není možné zobrazit pro takové skupiny statistiky citačního ohlasu publikací.

Jedním ze způsobů, jak umožnit zobrazování citačních statistik bylo naimplementovat vlastní systém, který chybějící funkcionalitu bude podporovat. Cílem mé práce bylo vybrat si dvě organizace, nastudovat si funkčnost jejich informačních systémů, nastudovat jejich aplikační rozhraní pro získání dat a vytvořit vlastní přehledný informační systém podporující chybějící funkcionalitu. Výsledný systém však neměl zobrazovat pouze součet citací pro daného autora, ale musel také pracovat s citacemi publikací bez vlastních citací autora, citace publikací bez citací všech autorů a také podle toho správně spočítat H-index autorů.

Hlavní část práce je rozvržena do osmi kapitol, které jsou seřazeny do větších myšlenkových celků a těmi jsou - analýza stávajícího řešení, analýza vlastního řešení, návrh řešení, popis implementace, ukázka formulářů aplikace a výsledná měření.

Analýza stávajícího řešení se skládá ze tří následujících kapitol. V kapitole 2 je rozveden popis obecné problematiky citací odborných publikací a způsob výpočtu Hirshova indexu. Pro získávání dat jsem zvolil informační systémy organizací Elsevier (IS Scopus) [1] a Clarivate Analytics (IS Web of Science) [2], které poskytují přístup k datům v rámci předplatného. V této kapitole jsou také popsány informační systémy organizací, které spravují informace o publikacích a jejich citacích.

Kapitola 3 obsahuje seznam funkcí, které měl výsledný systém implementovat. Tyto funkce byly zadány vedoucím diplomové práce. Dále je zde uveden odkaz na přílohu, která obsahuje návrhy formulářů. V kapitole 4 jsou již popsány způsoby, jakými informační systémy organizací poskytují data, a také konkrétní funkce API a IS Web of Science, které jsem využil při získávání citačních dat.

Analýzu vlastního řešení tvoří kapitola 5 Obecné algoritmy výpočtů. V této kapitole jsou slovně popsány algoritmy potřebné pro výpočet některých ukazatelů a ostatní algoritmy, které jsem využil při implementaci.

V kapitole 6 Návrh výsledného systému jsou uvedeny různé náhledy na výsledný systém z pohledu softwarového inženýrství jako je architektonický náhled nebo popis případů užití systému. Je zde také popsána databáze výsledného systému a dále jsou zde také popsány technologie s ukázkami použití při implementaci systému.

Kapitola 7 Popis implementace již popisuje způsoby plnění databáze získanými daty. Popisuje i implementaci přístupového objektu Scopus API včetně ukázek zdrojových kódů. Jelikož organizace Elsevier poskytuje jenom pro Scopus 11 různých aplikačních rozhraní, tak jsem v této kapitole popsal kroky nezbytné, abych získal potřebná data o citacích a publikacích dle autorů. Následně je v kapitole popsána implementace získávání dat z IS Web of Science a problém, na který jsem při této implementaci narazil. Také jsou zde uvedeny vybrané části funkcionality webové aplikace.

V kapitole 8 Webová aplikace jsou uvedeny ukázky uživatelského rozhraní výsledné aplikace a krátký popis funkčnosti z pohledu uživatele k jednotlivým ukázkám.

Poslední kapitolou práce je kapitola 9 Měření časů získávání dat, kde jsou uvedeny naměřené hodnoty časové náročnosti získávání dat, ale také porovnání optimalizace části implementace před a po úpravě.

Výsledný informační systém jsem zhotovil ve spolupráci s kolegou, jehož téma diplomové práce bylo zaměřeno na statistiky publikací a autorů samotných, proto jsou v mé práci uvedeny odkazy na reference, konkrétně na kolegovu diplomovou práci [3], ve které jsou blíže popsány některá úskalí a části výsledného systému.

2 Citace publikací a citační databáze

Mým úkolem bylo nastudovat problematiku publikací se zaměřením na jejich citační ohlas. Do té doby jsem neměl s touto problematikou žádné zkušenosti. Jako studijní materiál jsem zvolil kurz a články na webu ústřední knihovny VŠB-TUO [4, 5].

2.1 Citační analýza

Citační analýza je definována dle online kurzu [4] Mgr. Daniely Tkačíkové jako *matematicko-statistická bibliometrická metoda, která kvantifikuje vztahy mezi autory, dokumenty a vědními obory na základě bibliografických citací a bibliografických odkazů. Zkoumá citovanost dokumentů, četnosti citací v dalších pracích apod.*

Původně byla představována především impact faktorem, ale s rozvojem informačních systémů se rozšířila o další metriky umožňující hodnocení článků a autorů, např. o h-index.

V roce 2005 byl h-index představen americkým fyzikem Jorgem E. Hirschem. H-index se používá pro měření výkonnosti jednotlivých autorů, skupin autorů nebo celé instituce.

Ukazatel h-index je vypočítán ze seznamu publikací autora nebo skupiny autorů, pro kterou jej chceme zjistit. Definice h-indexu dle článku Mgr. Tkačíkové [5] je uvedena následovně. *Hodnota h-indexu se rovná počtu publikací (N) v seznamu, jež mají N nebo více citačních ohlasů.*

Výpočet h-indexu bez autocitací probíhá naprosto stejně jako výpočet klasického h-indexu, ale počítá se s hodnotami citací bez vlastních citací, případně s hodnotami citací bez autocitací všech autorů.

Dalším pojmem, kterým se v této práci také hodně zabývám, jsou citace bez vlastních citací (nebo citace bez autocitací). Autor odborného textu nemusí vycházet pouze z cizích článků, ale může také vycházet ze svých vlastních dřívějších článků, pak cituje své vlastní publikace. Také se může stát, že si autor bude citovat jiné své publikace, které s danou problematikou vůbec nesouvisí a chce si tak zvýšit svůj citační ohlas, protože to na první pohled vypadá, že se o jeho publikace opírá více článků. Proto databáze Scopus a Web of Science umožňují vyjmout tyto vlastní citace z výsledných hodnot. Stejně jako lze vyjmout vlastní citace, lze vyjmout i citace všech autorů, kteří se na dané publikaci podíleli. To znamená, že se vyjme z výsledného citačního ohlasu autora i citovanost ostatními autory v jejich publikacích, čímž se ještě více zpřesní hodnoty citačního ohlasu a předejde se tak započítání vlastních citací ostatních autorů.

Autoři odborných článků jsou často vědeckými pracovníky na univerzitách. Univerzity jako celky jsou děleny na jednotlivé části, kterými jsou fakulty, a ty se dále dělí na katedry. Této hierarchii také odpovídá rozdělení autorů do jednotlivých skupin, kde na nejvyšší úrovni je univerzita a nejnižší úroveň těchto skupin prakticky není omezena, protože autoři mohou být dále děleni do odborných skupin, které se také mohou dále dělit.

Protože jsou známy hodnoty počtů citací a h-indexů jednotlivých autorů v těchto skupinách, lze pak jednoduchým způsobem spočítat agregované hodnoty pro různé skupiny a úrovně těchto skupin autorů. Výsledné hodnoty pak slouží jako výkonnostní ukazatele publikační činnosti autorů v rámci skupin.

Poté, co jsem se seznámil s problematikou publikací a citací obecně, potřeboval jsem se seznámit i s organizacemi, které se touto problematikou zabývají. Zaměřil jsem se na dvě takové organizace a na jejich informační systémy - konkrétně na Scopus a Web of Science.

2.2 Citační databáze

Projekt Scopus [1] byl zahájen v listopadu 2004. Scopus je databáze abstraktů a citací recenzované literatury jako jsou vědecké časopisy, konferenční sborníky a knihy. Scopus nabízí nástroje pro sledování, analýzu a vizualizaci výzkumu. Poskytuje komplexní přehled výstupů výzkumu v oblasti vědy, techniky, medicíny, sociální vědy a humanitní vědy s uměním. Scopus obsahuje více než 66 milionů záznamů, z toho je:

- Více než 22 748 recenzovaných časopisů, ze kterých je více než 3 476 časopisů s otevřeným přístupem.
- Přes 558 titulů sériových neuzavřených knih, což představuje 34 000 svazků knih a tedy 1,3 milionů kusů.
- Přes 138 000 kusů uzavřených knih, ke kterým je každý rok přidáno dalších 20 000 kusů.
- 7,7 milionů konferenčních příspěvků.
- A 28 milionů patentů.

Více informací o Scopus je uvedeno v [6].

Protože Scopus indexuje takové množství publikací, tak dochází k časté shodě jmen jejich autorů, které je nezbytné odlišit. Často se také stává, že je jméno autora uváděno v jeho publikacích rozdílně. Například přehozením jména s příjmením, uvedením pouze příjmení a iniciálu jména apod. Proto Scopus automaticky přiřazuje nově uvedeným autorům **Scopus Author Id** [7, 8], ke kterému jsou další publikace se stejným jménem přiřazovány. Tímto automatickým přiřazováním identifikátoru se nemůže stát, že by autor, jehož publikace je ve Scopus indexována, neměl ID přiřazeno. Problémem je, že jsou jednomu autorovi pro různé verze jmen vygenerována

různá ID a tím je vytvořeno více profilů autora ve Scopus. Autor pak musí zadat požadavek pro sloučení těchto profilů. Indexování publikací probíhá buď skenováním, nebo automatickým dolováním dat z elektronických verzí publikací. I když algoritmus seskupení publikací k autorovi přihlíží k jeho jménu, instituci, oboru, adrese nebo datu citací, tak se v rámci instituce a stejných jmen autorů může stát, že jsou publikace autora přiřazeny jinému autorovi. Pak je na autorovi aby podal požadavek pro nápravu.

Druhou citační databází je Web of Science [2]. Přes jedno rozhraní lze za pomocí výkonných vyhledávacích a analytických nástrojů přistupovat k časopiseckým článkům, patentům, webovým stránkám a konferenčním sborníkům. Web of Science Core Collection je pečlivě vybraná a aktivně udržovaná bibliografická a citační databáze, kterou mnozí výzkumní pracovníci považují za velmi užitečnou ve svých oborech. Tato databáze obsahuje více než:

- 8 500 časopisů z přírodních věd,
- 3 000 časopisů ze společenských věd,
- 1 700 časopisů v oboru humanitních věd,
- 172 000 konferenčních sborníků,
- A 60 000 knih, ale tato databáze knih není pro VŠB-TU Ostrava přístupná.

Více obecných informací o Web of Science je uvedeno v [9] a informace o přístupném rozsahu pro VŠB-TU Ostrava jsou dostupné na webových stránkách univerzitní knihovny [10].

Ve Web of Science dochází s identifikací autorů ke stejnému problému s přiřazením publikací autorovi jako ve Scopus, ale ve větší míře. Web of Science nevytváří identifikátor autorovi automaticky podle jména uvedeného v publikaci, ale autor si po registraci ve Web of Science musí zažádat o vytvoření **ResearcherId** [8, 11] sám. Mnoho autorů si o vytvoření **ResearcherId** nezažádalo, proto nejsou uvedena ani u jejich publikací ve Web of Science.

Dalším způsobem identifikace autorů je **OrcId** [12]. V online kurzu [8] Mgr. Daniely Tkačíkové jsou cíle iniciativy **OrcId** uvedeny následovně. *Cílem iniciativy ORCID je vyřešení problémů s nejednoznačností osobních jmen v oblasti výzkumu a vědecké komunikace vytvořením centrálního registru jedinečných identifikátorů pro jednotlivé výzkumníky a rovněž otevřeným a transparentním spojovacím mechanismem mezi ORCID a dalšími existujícími systémy identifikace výzkumníků. Tyto identifikátory mohou být vzájemně propojeny s vědeckými výstupy výzkumných pracovníků s cílem posílit vědecký výzkum, efektivitu financování výzkumu i spolupráci v rámci vědecké komunity.* Identifikátor **OrcId** je globální a aktuálně jich je zaregistrováno 3 315 656 [13]. Podle **OrcId** lze vyhledávat autory ve Scopus i ve Web of Science, ale u publikací neregistrovaných autorů nejsou **OrcId** ve Web of Science uvedena stejně jako **ResearcherId**.

3 Požadované funkce systému

Zde uvádím seznam funkcí systému požadovaných vedoucím této diplomové práce, které se týkaly mé části práce. Jako první funkcí systému bylo vytvořit objekt pro komunikaci s databází Scopus. Objekt pro komunikaci s databází Web of Science měl za úkol vytvořit kolega v jeho diplomové práci [3]. Tyto objekty mají dále specifikovány funkce pro získávání dat z obou databází. Získaná data měla být uložena v databázi a následně měla být vizualizována ve webové aplikaci. Proto jsou od bodu 4 zmíněny požadované funkce na webovou aplikaci.

1. Získávání citačního ohlasu instituce.
    - Zde měly být získány souhrnné hodnoty počtu citací, počtu citací bez autocitací a h-indexu pro celou instituci.
2. Získávání citačního ohlasu skupin autorů, jako jsou fakulty, katedry nebo odborné skupiny.
    - Z API měly být získány hodnoty počtu citací, počtu citací bez vlastních citací, h-indexu a h-indexu bez vlastních citací. Počet citací bez autocitací měl být dále rozlišen na citace bez autocitací autora a citace bez autocitací všech autorů. Stejně měl být rozdělen i h-index bez autocitací.
3. Vytvoření databáze pro uchování informací o citacích, publikacích, autorech a jejich zařazením do skupin.
    - Databáze měla být naplněna z poskytnutého seznamu zaměstnanců.
    - Také mělo být vytvořeno ORM rozhraní pro komunikaci s databází.
4. Zobrazení obecného přehledu o publikacích a citacích autorů.
    - Po kliknutí na jméno autora měla stránka zobrazit jeho detailní informace.
5. Zobrazení statistik pro skupiny autorů získané ve funkci 2 a umožnit zde:
    - Výběr skupin v hierarchii a zobrazení citačních informací autorů vybrané skupiny i všech podskupin.
    - V případě, že se budou zobrazovat seznamy pro skupinu, která obsahuje další podskupiny, zobrazení i souhrnu pro jednotlivé podskupiny (v případě zobrazení seznamu pro celou fakultu se zobrazí i souhrn pro jednotlivé katedry).
    - Rozdělení souhrnné statistiky i do podskupin úrovně nižší o více než jednu úroveň.
    - Filtraci autorů dle jejich pracovního zařazení.
    - Pro autory, kteří již nemají pracovní nebo studentský úvazek na univerzitě, zobrazení citačních informací publikací publikovaných jen v době úvazku, ale umožnit také uživateli zahrnout do výsledků kompletní publikační činnost autora.

- Zobrazení detailních informací autora po kliknutí na jeho jméno.
- Seřazení seznamu autorů dle všech zobrazených atributů.
- U tabulky s autory stránkování s možností výběru velikosti stránky: 20, 50, 100, 200.

6. Vytvoření administrační část aplikace a umožnit zde:
   - vytváření, úpravu a mazání skupin v jejich stromové struktuře,
   - přidávání, úpravu a mazání pracovních zařazení autorů,
   - přidání nového autora a jeho úpravu včetně zařazení do skupiny.
     - Při vytváření nového autora zobrazit při vepsání Scopus ID počet publikací autora v IS Scopus a stejně při vepsání OrcID a ResearcherID zobrazit počet publikací z Web of Science.
     - Při přidávání autora nastavit autorovi i rok počátku úvazku.
     - Umožnit filtrovat a seřazovat seznam autorů.
     - Umožnit deaktivaci autora s určením roku ukončení úvazku na univerzitě.

Získávání dat o instituci jako celku nebylo možné vytvořit, ale lze tyto informace získat jako hodnoty pro skupinu všech autorů instituce. Funkce 4 tak překrývá přidanou hodnotu funkce 3, proto funkci 3 již v práci dále nepopisuji.

Abych při vývoji webové aplikace nevytvářel něco jiného, než po mně bylo vyžadováno, museli jsme si s vedoucím práce ujasnit představu o tom, jak by měla výsledná aplikace vypadat. Navrhl jsem tedy několik formulářů v nástroji Visio od firmy Microsoft a odeslal vedoucímu diplomové práce, návrhy jsou uvedeny v příloze A.

4 Funkčnost Scopus API a Web of Science

Kapitola je rozdělena na 3 podkapitoly. V první podkapitole je popsána funkčnost Scopus API a možnosti získávání dat, ale také proces nezbytný pro povolení jednoho ze Scopus API a nepřesnost dat získaných z jiného API. V druhé podkapitole je popsán IS Web of Science, ze kterého lze získat citační a publikační informace. Zde je také popsána nejvhodnější možnost získávání dat včetně úskalí, na která jsem narazil. V poslední podkapitole jsou shrnuta a popsána data, která API a Web of Science poskytují, ale také citační data, která z těchto systémů nelze získat.

4.1 Scopus REST API

Organizace Elsevier poskytuje pro Scopus 11 různých aplikačních rozhraní, skrze která lze získat data prakticky o všech citacích a abstraktech, která jsou indexována v IS Scopus.

Všechna Scopus API jsou implementována RESTful architekturou, proto komunikace s nimi probíhá skrze HTTP požadavky a odpovědi. Přímo v URL adrese požadavku je specifikován výčet atributů pro navrácení z API a také filtry pro omezení vráceného výsledku.

Pro přístup ke Scopus API je nezbytné si nechat přidělit klíč k API. Pro získání klíče jsem se musel nejdříve zaregistrovat, poté mi webová aplikace s dokumentací k API [14] vygenerovala a přidělila klíč - jedná se o 32místný kód složený kombinací čísel a písmen. Klíč jsem nadále připojoval jako součást hlavičky HTTP požadavku na API. Klíč může získat kdokoliv a může i používat Scopus API bezplatně, ale musí dodržovat zásady používání API. Avšak plný přístup k API je poskytován pouze klientům, kteří mají předplatné pro Scopus. Klienti bez předplatného mají přístup pouze k omezeným základním metadatům o publikacích a citacích.

Z jedenácti API, které organizace Elsevier poskytuje pro Scopus, potřebná data vracela 3 API, a to konktrétně Scopus Search API, Author Retrieval API a Citations Overview API. V požadavku na API lze specifikovat, v jakém formátu má být navrácená odpověď vygenerována. U Scopus Search API a Author Retriaval API jsou to možnosti:

- application/json,
- application/atom+xml,
- application/xml.

U Citations Overview API jsou to možnosti:

- application/json,
- text/xml,
- application/xml.

Při implementaci funkce **2** uvedených v kapitole 3 jsem využíval Author Retrieval API pro získání h-indexu a počtu citací autora, ale při zpracování dat z API mi neseděly některé počty

citací, ale ani h-indexy pro autora. Hodnoty se vždy o trochu lišily oproti IS Scopus. Hledal jsem chybu ve své implementaci, ale když jsem si byl jistý, že chyba není na mé straně, napsal jsem email na podporu organizace Elsevier. Organizace sídlí v Nizozemí, proto celá komunikace s podporou probíhala v anglickém jazyce. Od podpory jsem dostal následující odpověď.
Hi Jiří,

Thanks for your email.

The author metrics on scopus.com are calculated on the fly whereas the author metrics that are being retrieved by api.elsevier.com are pre-calculated for the APIs. The database used for that purpose is updated every 2 to 6 weeks which is why those values may lag behind Scopus.

In addition, that database only uses the content which has been published from 1996 onwards. There are historical reasons for that and we intend to change that, but it will require a major back end effort and I don't have a timeline for it.

Note that this is only valid for author metrics; all other content and values between Scopus and APIs are the same.

Best, Meshna

Meshna Koren
Integration Support Coordinator Product Management Platform and Data Integration, Research Products
Elsevier BV Radarweg 29, Amsterdam 1043 NX, The Netherlands m.koren@elsevier.com

Byť je v odpovědi uvedeno, že databáze pro API je aktualizována každých 2 až 6 týdnů, ani po 6 týdnech nebyla data aktualizována.

Abych mohl v dávkách odesílat publikace pro získání jejich citací, potřeboval jsem správný počet publikací pro autora, proto jsem se na konec rozhodl využít Scopus Search API, které již vrací správný počet publikací. Ukazatele h-index jsem také původně získával z Author Retrieval API, ale v tuto chvíli je po aktualizaci dat ve vlastní databázi vždy spočítá aplikace a uloží zpět do databáze. Proto jsem s Author Retrieval API dále při implementaci nepracoval a nakonec jsou využita pouze 2 API.

Scopus Search API reprezentuje vyhledávání napříč Scopus. API umožňuje pro vyhledání určit i booleovské dotazy do Scopus indexu a vrací relevantní metadata v předem určeném formátu, který jsem již uvedl. API vrací velké množství informací o publikacích, jako jsou identifikátory publikace, odkaz na abstrakt publikace, název publikace, datum vydání, seznam autorů publikace a další.

Já jsem API využil tak, že jako dotaz je odesláno Id autora, pro kterého jsou navrácena ScopusId publikací, a ty jsou následně odesílána na Citations Overview API pro získání jejich citací. Parametrem předaným Scopus Search API může být i seznam ScopusId publikací, pro které mají být vráceny informace. API vrací také celkový počet publikací autora, který jsem využil při implementaci funkce **6**. Počet publikací je zobrazen administrátorovi aplikace při přidávání nového autora.

Citations overview API reprezentuje rozhraní pro získání počtu citací dokumentů členěné dle let s možností vyloučit autocitace, ale také poskytuje souhrnné informace pro jednotlivé publikace, proto jsem tohoto API využil při implementaci získávání všech citačních dat ve funkcích **1** a **2**. API vrací výsledek ve struktuře, která skoro přesně odpovídá struktuře tabulky v IS Scopus včetně hlaviček tabulky (viz obrázek 1). Pro porovnání je v příloze B uveden XML dokument.

Documents

Citations

Sort on: Date (newest) Citation count (descending) ...

			<2012	2012	2013	2014	2015	2016	Subtotal	>2016	Total
		Total	13	4	12	1	3	0	20	0	33
1	Cost-based holistic twig joins	2015							0		0
2	QuickDB - Yet another Database Management System?	2014							0		0
3	QuickXDB: A prototype of a native XML DBMS	2013							0		0
4	Optimal and efficient generalized twig pattern processing: A...	2013					1		1		1
5	XML query processing - Efficiency and optimality	2012							0		0
6	Multidimensional implementation of stream ADT	2012							0		0
7	Fast decoding algorithms for variable-lengths codes	2012			7	1	2		10		10
8	Index-based N-gram extraction from large document collection...	2011							0		0
9	Flexible cache for a database management systems	2011							0		0
10	On support of ordering in multidimensional data structures?	2010							0		0
11	On support of ordering in multidimensional data structures	2010		1					1		1
12	Optimization of disk accesses for multidimensional range que...	2010	1	1					1		2
13	Benchmarking the compression of XML node streams	2010	1	1					1		2

Obrázek 1: Přehled citací v IS Scopus

S Citations Overview API nastaly komplikace. Dle dokumentace na webu organizace Elsevier mělo být rozhraní povoleno, proto jsem opakovaně zkoušel upravovat a odesílat požadavky, ale server vždy vrátil chybu 500 – Internal Server Error, která mi toho moc nenapověděla.

Po dalších neúspěších jsem napsal email na podporu organizace Elsevier. Zaměstnanec podpory mě opakovaně odkazoval na podmínky používání API a případně na jiná API, která však nevracela informace, které jsem ke splnění svého úkolu potřeboval. Až jsme se se zaměstnancem domluvili a on nakonec pochopil, že se opravdu bez tohoto API neobejdu, tak jsem se od něj dozvěděl, že pro povolení API potřebovali schválení pro použití osobou odpovědnou za Scopus předplatné pro univerzitu.

Zjistil jsem, že odpovědnou osobou za předplatné byla Mgr. Daniela Tkačíková, v té době ředitelka Ústřední knihovny VŠB-TUO, tak jsem se za ní zastavil osobně, vysvětlil jsem jí, co dělám a požádal jsem ji o schválení použití API. Paní Tkačíková mi vyhověla, napsala na podporu organizace Elsevier a oni následně toto API povolili pro můj API klíč.

Také jsem musel potvrdit, že Citations Overview API bude používáno v souladu s politikou IR/CRIS [14] a hlavně, že se budu dotazovat na počty citací článků, které byly publikovány autory naší instituce, a že tyto počty nebudou zobrazeny veřejně na webu.

Po schválení a povolení API jsem mohl na získávání dat dále pracovat. Od prvního emailu na podporu až po povolení API uběhlo 2 a půl měsíce.

4.2 Web of Science Core Collection

Organizace Clarivate Analytics pro IS Web of Science také poskytuje API, skrze které se dají získat data o publikacích, ale s kolegou jsme toto API použít nemohli, o čemž píše více kolega ve své práci [3]. V tomto případě bylo zapotřebí vyřešit nastalou otázku - jak získat data z Web of Science?

Odpověď na otázku je - získávat data přímo z webové aplikace Web of Science. Protože se jedná o běžnou webovou aplikaci dostupnou skrze protokol HTTP, lze s ní pomocí tohoto protokolu komunikovat.

Získávat data skrze HTTP požadavky pouze s metodou *GET* není možné kvůli URL adrese, která vypadá následovně 1.

```
"https://apps.webofknowledge.com/Search.do?product=WOS&SID=4DlH1W3BQMyowlmv8PY&
    search_mode=GeneralSearch&prID=d80ba6f9-a63a-4d6d-9499-61171f8e04bb"
```

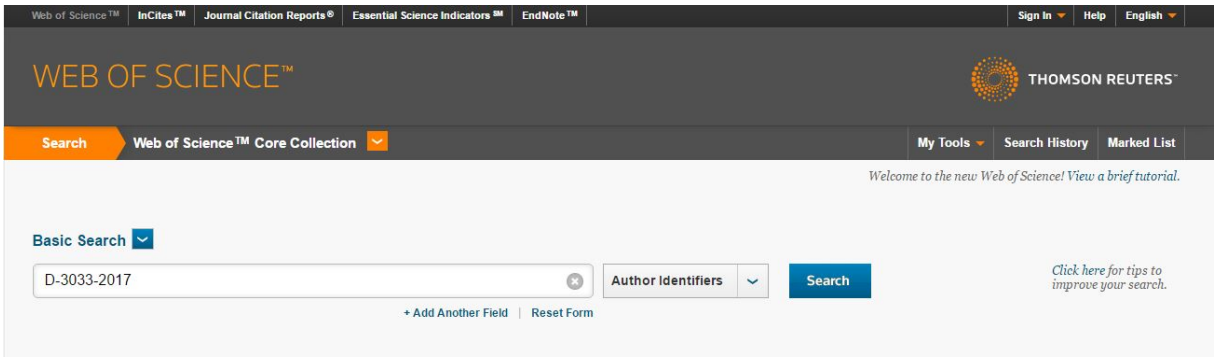
Výpis 1: URL adresa Web of Science

V URL jsou vidět dva vygenerované klíče **SID** a **prID**. Při vyhledání publikací se neskládají parametry vyhledávání do URL viditelně, ale IS vygeneruje klíč **SID**, se kterým jsou parametry

pro vyhledání publikací uchovány na serveru. Proto není možné generovat URL adresy pro získání HTML s publikacemi vybraného autora.

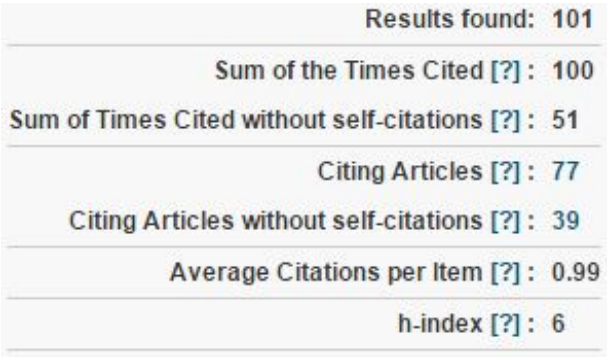
Klíč **SID** také slouží k identifikaci **Session** přihlášeného uživatele. Tato **Session** má svou životnost. V případě, že je uživatel v IS aktivní, tak se mu **Session** obnoví se stejným klíčem, aniž by si toho všiml, ale v případě že by s IS Web of Science neprovedl žádnou interakci, **Session** by se ukončila a pokud by chtěl dále s IS pracovat, IS by vygeneroval novou, ovšem jen v případě, že uživatel přistupuje k Web of Science ze sítě se zaplaceným předplatným, jinak by IS uživatele přesměroval na přihlášení.

Výchozí stránkou Web of Science je stránka pro vyhledání. Pro potřeby vyvíjeného systému bylo nezbytné vyhledat autorovy publikace a jejich citační ohlas. Aby byly výsledky vyhledávání publikací přesné, je lepší vyhledávat dle ResearcherID nebo OrcID autora viz obrázek 2.



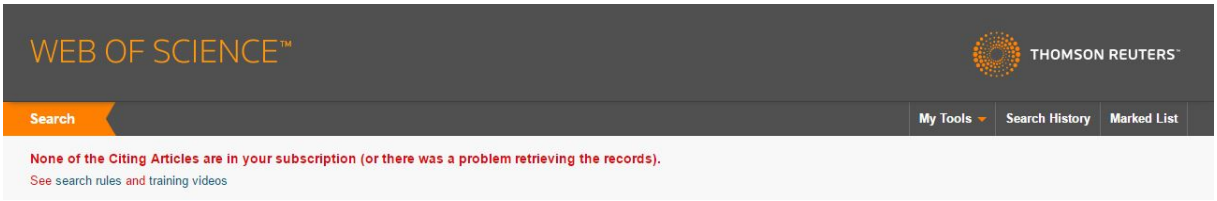
Obrázek 2: Výchozí vyhledávací stránka Web of Science

Po vyhledání dle ID autora IS zobrazí stránku se stránkovaným seznamem publikací. Zde je zobrazen i celkový počet publikací autora, který jsem využil, stejně jako u Scopus API, pro jeho zobrazení při přidávání nového autora ve funkci **6**. Na této stránce také IS umožnuje vytvořit citační report 3, kde jsou uvedeny základní informace o citacích, jako je počet citací, počet citací bez autocitací nebo h-index. Zde však nejsou uvedeny hodnoty počtů citací bez autocitací všech autorů ani h-index bez autocitací, proto jsem této možnosti nevyužil.



Obrázek 3: Citační report ve Web of Science

Web of Science umožňuje uživateli zobrazit i detailní stránku publikace (viz obrázek 5), kde jsou k dispozici i ID všech autorů, pokud je všichni autoři vyplnili u svého profilu. Napravo od publikace je uveden počet citací publikace. Když je počet citací větší než 0, pak samotný počet je uveden jako odkaz, po jehož odkázání je zobrazen seznam citujících publikací. Zde jsem narazil na problém, že jsem nemohl zobrazit citující publikace pro jednoho z autorů a IS Web of Science mi zobrazil chybu, která je vidět na obrázku 4. Tato chyba je s vysokou pravděpodobností způsobena tím, že škola má omezeno předplatné a pro zobrazení citujících publikací by muselo být předplatné rozšířeno.



Obrázek 4: Chyba zobrazení citujících publikací

Detailní stránka umožňuje také přechod na další publikaci v pořadí. Tak je možno projít programově všechny publikace. Tuto stránku IS jsem potřeboval využít při řešení funkce **2** v kapitole 3 právě kvůli uvedeným ID autorů.

Obrázek 5: Detail publikace ve Web of Science

4.3 Shrnutí poskytovaných ukazatelů z Web of Science a Scopus API

V předchozích podkapitolách jsou uvedeny ukazatele, které jednotlivá Scopus API a Web of Science poskytují. Pro lepší přehlednost jsou zde tyto ukazatele shrnuty do 2 tabulek rozdělených podle zdrojů. V obou tabulkách jsou uvedeny stejné ukazatele pro lepší srovnání poskytovaných funkčností mezi zdrojovými databázemi.

V tabulce 1 jsou ukazatele ještě rozděleny dle jednotlivých API, ze kterých je lze získat. Jak jsem již uvedl, Scopus nabízí více API a ty poskytují mnohem více ukazatelů a funkčností, které zde nejsou popsány, protože jsem je ve své práci nevyužil. Více informací k poskytovaným API a jejich funkčnostem je uvedeno v dokumentaci k API [14]. Také jsem již uvedl, že Author Retrieval API jsem ve výsledné aplikaci nepoužil kvůli nepřesným výsledkům, proto bylo nezbytné doimplementovat výpočet citačních ukazatelů pro autora z citačních informací jeho publikací.

Tabulka 1: Scopus - Porovnání poskytovaných funkčností

API	**Ukazatel**	**Poskytují**	**Využil jsem**	**Nutno doimplementovat**
Scopus Search API	Počet publikací autora	✓	✓	
	Seznam publikací autora	✓	✓	
	Detail publikace včetně autorů a jejich ID	✓	✓	
Citations Overview API	Počet citací publikace	✓	✓	
	Počet citací publikace bez autocitací autora	✓	✓	
	Počet citací publikace bez autocitací všech autorů	✓	✓	
Author Retrieval API	Počet citací autora	✓		✓
	Počet citací autora bez autocitací	✓		✓
	H-index	✓		✓
	H-index bez autocitací	✓		✓
	Počet citací autora bez autocitací všech autorů			✓
	H-index bez autocitací všech autorů			✓

Stejným způsobem jako u Scopus API jsou vypočítávány ukazatele pro autora z dat Web of Science. Z citačního reportu lze získat základní citační informace autora. Zbylé ukaza-

tele je však nezbytné vypočítat z citačních informací publikací, čímž jsou získány stejné hodnoty ukazatelů autora jako z citačního reportu a samotný výpočet ukazatelů h-index a počet citací bez autocitací je časově méně náročný, než jejich získávání z citačního reportu.

Tabulka 2: Web of Science - Porovnání poskytovaných funkčností

Ukazatel	**Poskytují**	**Využil jsem**	**Nutno doimplementovat**
Počet publikací autora	✓	✓	
Seznam publikací autora	✓		
Detail publikace včetně autorů a jejich ID	✓	✓	
Počet citací publikace	✓	✓	
Počet citací publikace bez autocitací autora			✓
Počet citací publikace bez autocitací všech autorů			✓
Počet citací autora	✓		✓
Počet citací autora bez autocitací	✓		✓
H-index	✓		✓
H-index bez autocitací			✓
Počet citací autora bez autocitací všech autorů			✓
H-index bez autocitací všech autorů			✓

V porovnání poskytovaných funkčností obou zdrojových databází poskytují Scopus API více ukazatelů než je tomu u IS Web of Science. Z funkcí požadovaných vedoucím diplomové práce nebylo možné získat počet citací autora bez autocitací všech autorů ani z jedné z databází, stejně tak nebylo možné získat h-index bez autocitací všech autorů. Pro umožnění zobrazení těchto ukazatelů bylo nezbytné naimplementovat jejich výpočet.

5 Obecné algoritmy výpočtů

Aby aplikace splňovala požadavky na funkčnost výsledného systému, některé ukazatele musely být vypočítány, protože nešly jednoduše získat z API systémů organizací 4. Obecné algoritmy těchto výpočtů v této kapitole popisuji.

5.1 Počet citací bez autocitací autora z Web of Science

Protože jsou data získávána přímo z webu Web of Science, bylo nezbytné počet citací bez autocitací autora vypočítat. Pro výpočet ukazatele je potřeba seznam publikací, jejich citací a k publikacím i seznam jejich citujících publikací.

Hlavní myšlenkou algoritmu je odečtení 1 citace z celkového počtu citací publikace v případě, že existuje shoda citující publikace s některou z publikací autora.

Protože je potřeba výpočet provést pro každou publikaci autora, minimální počet vnořených cyklů v algoritmu je 3. Jeden pro průchod publikacemi, druhý pro průchod citujícími publikacemi a třetí pro opětovný průchod publikacemi autora.

5.2 Počet citací bez autocitací všech autorů z Web of Science

Pro výpočet počtu citací bez autocitací všech autorů je nezbytné mít získány navíc seznamy všech autorů pro publikace i pro jejich citující publikace.

Hlavní myšlenkou tohoto algoritmu je odečtení 1 citace od celkového počtu citací publikace v případě, že existuje shoda některého z autorů publikace s některým z autorů citujících publikací, ale nesmí se stát, že by byla 1 citace (tudíž 1 citující publikace) odečtena dvakrát kvůli shodě více autorů u stejných publikací.

Implementace tohoto algoritmu vyžaduje minimálně 4 vnořené cykly k průchodu publikací, jejich autorů, citujících publikací a jejich autorů. Ale procházení citující publikace musí být ukončeno v případě nalezení shody autorů.

5.3 H-indexy

Pro výpočet h-indexu je nezbytné seřadit seznam publikací sestupně podle počtu citací. Poté je h-index roven pořadovému číslu publikace, která má počet citací větší nebo roven tomuto pořadovému číslu.

V implementaci výpočtu h-indexu využijeme nejméně 3 cykly. 2 vnořené cykly pro seřazení hodnot a jeden cyklus pro průchod seřazenými hodnotami a výběr výsledku.

5.4 Agregace ukazatelů

Jedním z hlavních požadavků byly souhrnné statistiky citací pro skupiny autorů. Pro potřeby těchto agregací citačních dat vystačí funkce pro součet a aritmetický průměr.

Výpočet souhrnných hodnot citačních ukazatelů s sebou nese několik úskalí. Počet citací autora, počet citací bez autocitací autora i počet citací bez autocitací všech autorů nesmí být sečten jednoduše pro seznam autorů, protože by se započítávaly počty citací některých publikací duplicitně. Jsou to publikace, na kterých se podílelo více autorů ze stejné skupiny. Zde uvedu příklad.

V případě souhrnných hodnot pro katedru telekomunikační techniky, by se započítala publikace *Utilizing the Neural Networks for Speech Quality Estimation Based on the Network Characteristics* minimálně dvakrát, protože se na ni podíleli autoři - doc. Ing. Miroslav Vozňák, Ph.D. a Ing. Jan Rozhon, Ph.D. Pro každého z autorů by již byl počet citací této publikace započítán při výpočtu počtu citací pro jednotlivce a oba spadají pod uvedenou katedru. Takových publikací je i v tomto případě mnohem více a tato chyba by se vyskytovala často, protože často na publikacích spolupracují autoři ze stejných kateder.

Také není vhodné sčítat ukazatel h-index autorů dané skupiny z podobného důvodu, jako tomu bylo u počtu citací. H-index je vypočítán z počtu citací publikací autora, jenže opět se na publikacích podílelo více autorů, a tak jsou počty citací některých publikací již započítány v h-indexech více autorů, proto by nebyl správným ukazatelem součet těchto h-indexů.

5.5 Průchod stromové struktury

Pro průchod stromové struktury jsem využil preorder algoritmus průchodu zleva doprava. Jedná se o velmi jednoduchý algoritmus obsahující pouze akci, která se má provést s daným uzlem a jeden cyklus pro průchod potomků uzlu, v němž je rekurzivně volána sama funkce s parametrem potomka. Algoritmus je nazančen pseudokódem v ukázce 2.

```
void TreePassage(node):
   do action with node
   foreach child in node.ChildNodes do
      TreePassage(child)
```

Výpis 2: Pseudokód algritmu průchodu stromové struktury

6 Návrh výsledného systému

V této kapitole je popsána architektura celého systému a struktura přístupových objektů k citačním databázím. Následuje ukázka vlastní databáze s krátkým popisem jednotlivých tabulek a také jsou zde popsány jednotlivé technologie využité při implementaci systému společně s jejich ukázkami. Dle požadavků vedoucího práce, byla aplikace rozdělena na dvě části podle uživatelských rolí. Formuláře pro zobrazení informací o publikacích a citacích a administrační formuláře určené jen uživatelům v roli administrátora pro správu a aktualizaci všech dat v aplikaci.

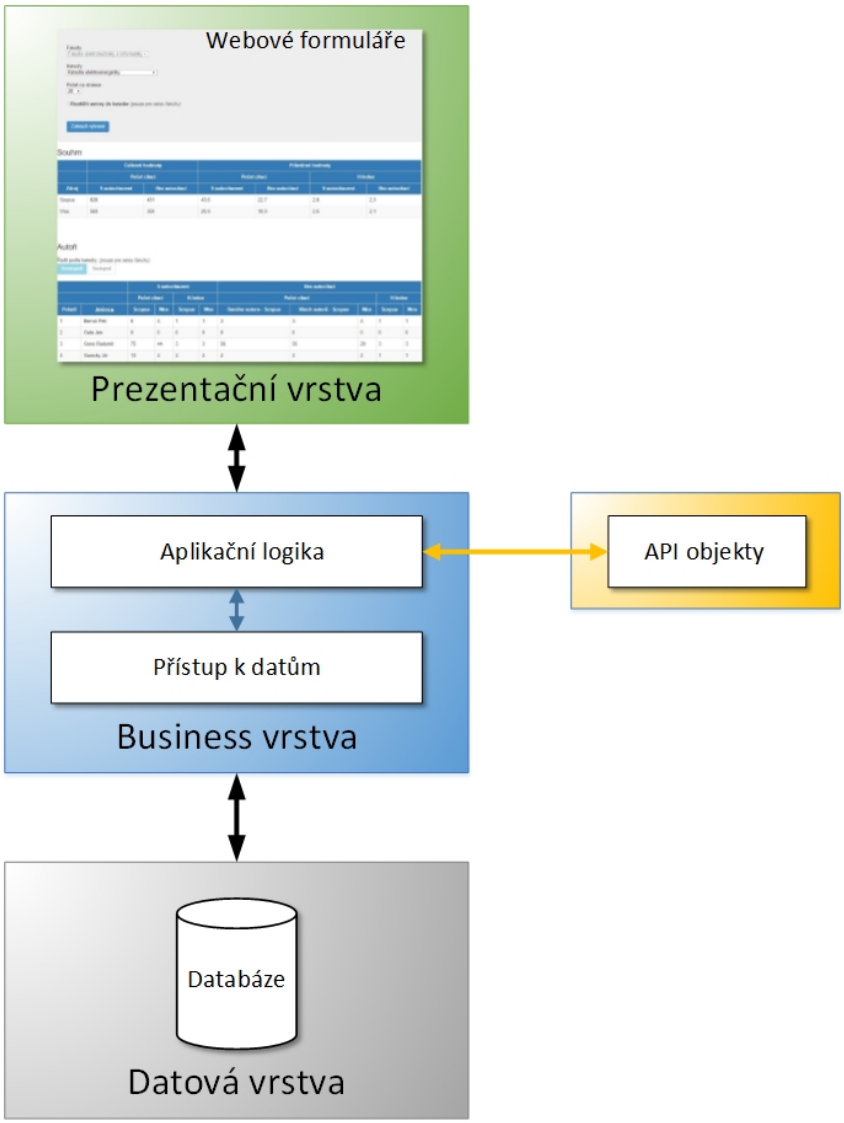
6.1 Architerktura systému

Jako u vsěch webových aplikací je i fyzická architektura této aplikace navrhnuta jako klient-server a data aplikace jsou přenášena po síti pomocí HTTP protokolu. Z logického pohledu je v aplikaci použita 3vrstvá architektura, která se skládá z prezentační vrstvy, business vrstvy a datové vrstvy. Datovou vrstvu tvoří SQL databáze, která je popsána v následující podkapitole, prezentační vrstvu tvoří webové formuláře včetně jejich zdrojového kódu, kde se tvoří a plní jednotlivé prvky uživatelského rozhraní. Business vrstva je složena z objektů pro přístup k datům, jejich mapování do objektů entit a z aplikační logiky, která zpracovává data a provádí výpočty ukazatelů h-index.

V této vrstvené architektuře zobrazené na obrázku 6 je rozdělena zodpovědnost za různé funkčnosti jednotlivým vrstvám. Konkrétně třídy webových formulářů, které se nacházejí v prezentační vrstvě, mají za úkol řešit pouze inicializaci, úpravu a plnění ovládacích prvků stránky. O volání metod pro přístup k datům, výpočet ukazatelů a další operace s daty v rámci aplikace je již postaráno třídami s názvem končícím *Controller*, které tvoří aplikační logiku. Toto rozdělení zodpovědnosti vede k lepší údržbě a čitelnosti zdrojového kódu.

Třívrstvé architektuře odpovídá také fyzické rozdělení vrstev na různé servery. SQL databáze datové vrstvy je umístěna na serveru **dbsys.cs.vsb.cz\STUDENT**, kde je nainstalován SQL Server 6.4.3, zatímco aplikační logika a objekty pro přístup k datům jsou umístěny na serveru **dbedu.cs.vsb.cz**.

Přístupové objekty k API jsou reprezentovány knihovnami tříd, ze kterých jsou přes rozhraní volány metody pro získání dat. Tyto metody jsou volány z aplikační logiky webové aplikace, která je předá vrstvě přístupu k datům a ta je uloží do databáze. Po vložení dat jsou data opět vybrána z databáze a v aplikační logice jsou spuštěny funkce pro výpočet h-indexů a jejich uložení do databáze.



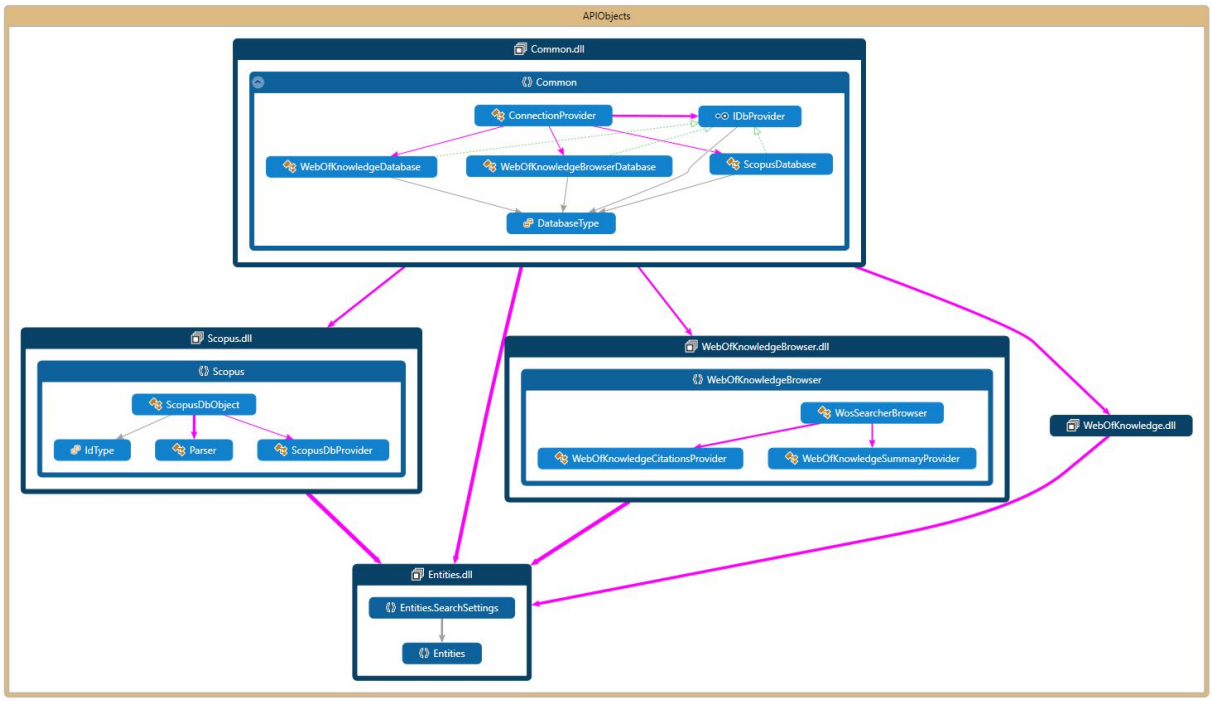
Obrázek 6: Architektura systému

6.2 Struktura přístupových objektů k API

Architekturu přístupových objektů vymyslel kolega a také ji ve své práci [3] více rozepisuje, ale já jsem do této struktury projektů vytvářel svůj projekt a třídy s funkcionalitou pro získávání dat, proto zde také stručně popíši její rozvržení. Na obrázku 7 je viditelné rozdělení projektů včetně jejich tříd.

Projektů pro přístup k databázím je celkem 5:

- Common
- Scopus
- WebOfKnowledgeBrowser



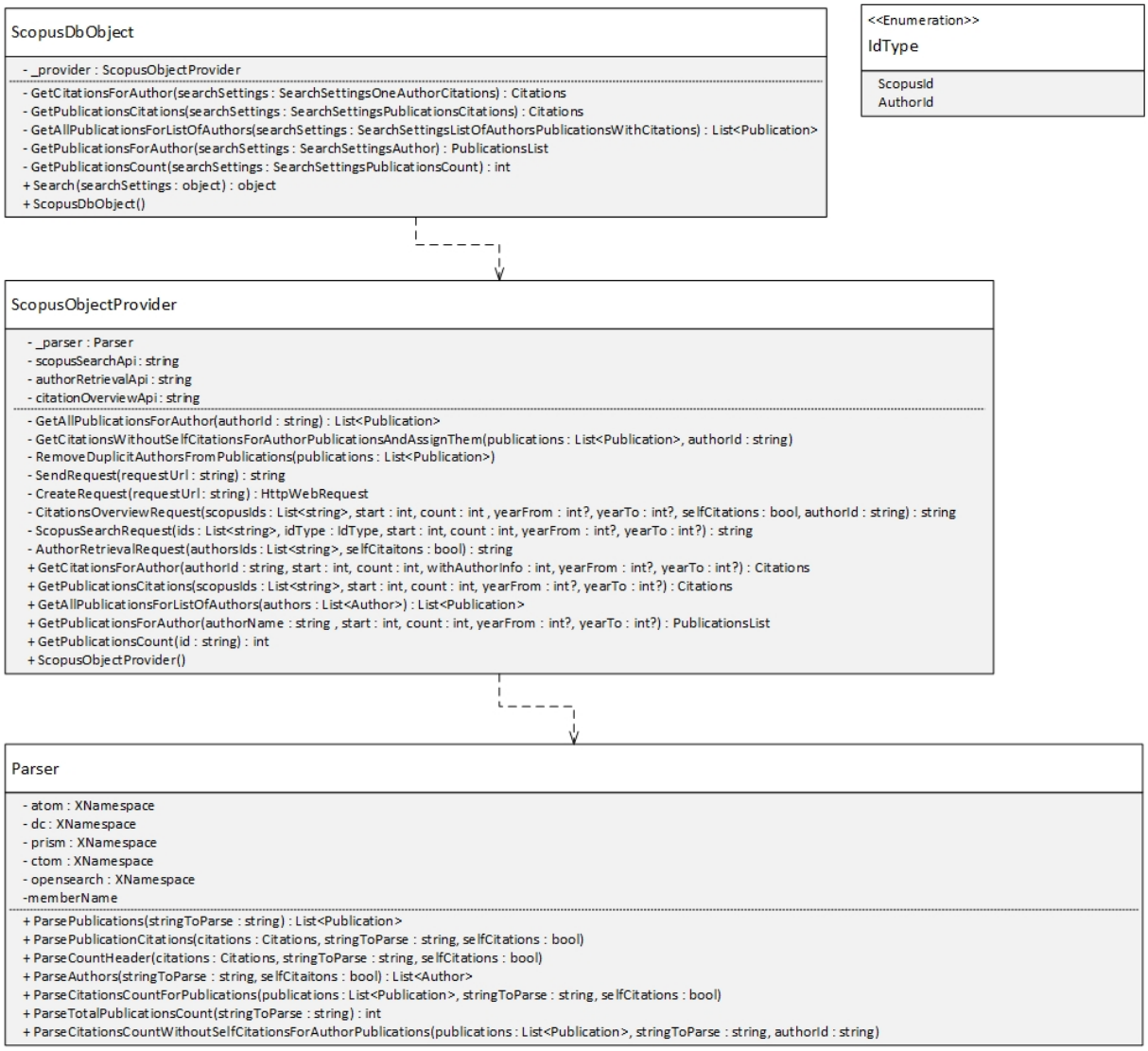
Obrázek 7: Architektura projektů

- WebOfKnowledge
- Entities

Projekt **Common** slouží jako zprostředkovatel pro práci s přítupovými objekty. V projektu **Common** se nachází rozhraní *IDbProvider*, které funguje jako obal pro přístupové objekty. Nachází se zde také třída *ConnectionProvider*, která slouží jako zprostředkovatel připojení k jednotlivým databázím skrze rozhraní. V projektu jsou dále 3 třídy, které dědí z uvedeného rozhraní. Každá ze tříd odpovídá jednomu z přístupových objektů (databází). V těchto třídách je volána jednotná metoda pro získání dat, jejíž způsob získávání dat a návratová hodnota je určena pomocí nových objektů specifického jména.

Díky této architektuře je možné v budoucnu bez problémů přidat nový projekt pro přístup k další databázi a to tak, že po přidání nového projektu s implementací se pouze přidá do projektu **Common** nová třída, která bude implementovat rozhraní *IDbProvider* a ve třídě *ConnectionProvider* se přidá do metody *Connect* případ pro vytvoření nového objektu předem popsané třídy.

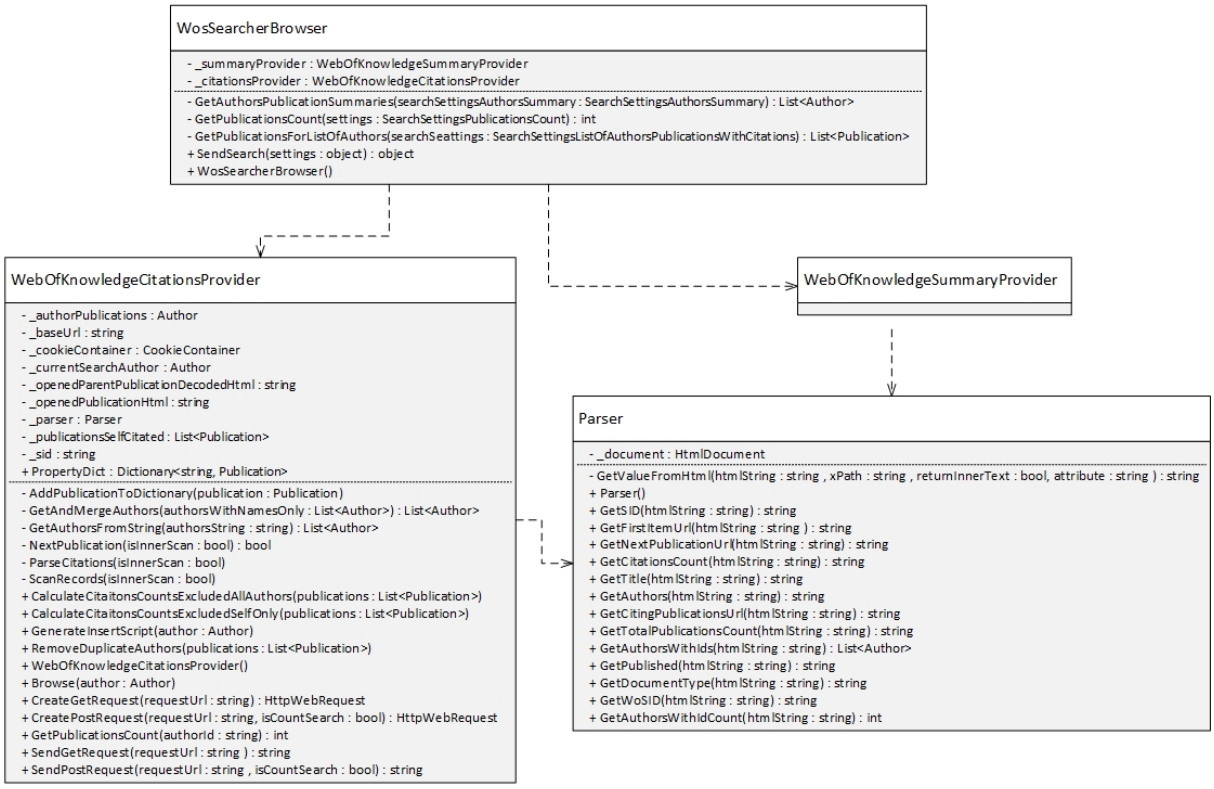
Dalším projektem v přístupových objektech je projekt **Scopus**, ten obsahuje samotný přítupový objekt ke Scopus API. Nacházejí se zde 3 třídy - *ScopusDbObject*, *ScopusObjectProvider* a *Parser*. Třída *ScopusDbObject* pracuje jako prostředník, který rozhoduje, která metoda ze třídy *ScopusObjectProvider* bude volána, a tedy která data budou získávána. Ve třídě *Parser* se nacházejí metody pro parsování výsledného XML dokumentu do objektů entit. Třídní diagram přístupového objektu ke Scopus API je vidět na obrázku 8.



Obrázek 8: Třídní diagram přístupového objektu ke Scopus API

Struktura projektu **WebOfKnowledgeBrowser** je totožná jako struktura projektu **Scopus**. Také rozdělení metod do tříd a jejich použití je totožné. Jen je zde jedna třída navíc s názvem *WebOfKnowledgeSummaryProvider*, která slouží kolegovi pro získávání dat o publikacích. Kolega o této třídě více píše ve své práci [3], proto jsem neuváděl ani její atributy a operace v třídním diagramu 9.

Jedním z projektů pro přístup k API je i projekt *Entities*. Projekt obsahuje třídy s vlastnostmi ve stromové struktuře pro uchování skupin autorů, publikací a dalších. Na projektu **WebOfKnowledge** jsem nepracoval a ve výsledné aplikaci se nepoužívá, proto jej zde nepopisuji vůbec.



Obrázek 9: Třídní diagram přístupového objektu k Web of Science

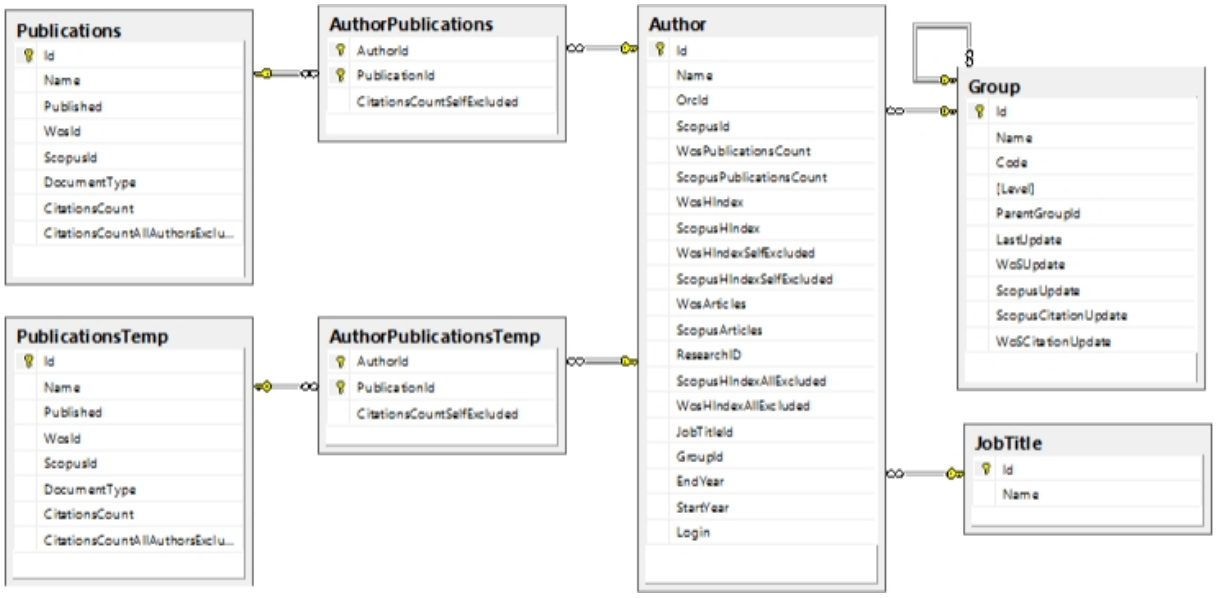
6.3 Databáze

Do doby, než jsem obdržel seznam autorů Fakulty elektrotechniky a informatiky, fungovala aplikace bez databáze a data, která zobrazovala, získávala v reálném čase ze Scopus API. Následně, dle požadavku **3** kapioly 3, jsem vytvořil databázi s autory a jejich zařazením do kateder, se kterými aplikace pracuje. Pro ukládání dat jsem použil Microsoft SQL Server, na kterém jsme se dohodli s kolegou.

Protože jsme s kolegou nepoužili žádný nástroj pro verzování databáze a neuchovávali jsme SQL skripty v TFS[1], stalo se, že jsme měli odlišné verze databáze, proto jsme se museli dohodnout na konečné společné verzi. V následujícím obrázku 10 je vidět schéma výsledné databáze.

Protože jsou získávána data ze dvou různých publikačních a citačních bází dat, musel jsem se při tvorbě schématu databáze také zamyslet, kam uložit publikace s citacemi ze Scopus a kam z Web of Science. Slučovat počty citací z různých zdrojů porovnáním textových řetězců s názvy publikací do jedné entity v databázi jsem odmítl. Proto jsem uvažoval, zda vytvořit zvlášť tabulku pro publikace ze Scopus a druhou pro publikace z Web of Science, ale protože by měly tyto dvě tabulky naprosto stejné atributy, rozhodli jsme se s kolegou vytvořit pouze jednu, kde se ukládají data z obou zdrojů a jsou rozlišitelné podle toho, jestli nemají atributy **ScopusId** nebo **WosId** hodnotu **NULL**.

[1]Námi používané TFS dostupné na URL adrese `https://tomasjirka.visualstudio.com/SemestralProject`



Obrázek 10: Finální schéma databáze

V dalších odstavcích krátce popisuji tabulky a atributy, které jsem potřeboval pro svou práci, informace o zbylých tabulkách a atributech jsou v kolegově diplomové práci [3].

Author

Tabulka autor uchovává informace o autorovi.

- Id,
- Name - celé jméno autora,
- OrcId - Open Researcher and Contributor ID, slouží pro vyhledání publikací autora na Web of Science,
- ResearchId - identifikátor přidělený autorovi ve Web of Science, také slouží pro vyhledání publikací autora na Web of Science,
- ScopusId - Id přidělené v IS Scopus,
- WosHIndex - h-Index včetně vlastních citací vypočítaný z dat získaných z Web of Science,
- ScopusHIndex - h-Index včetně vlastních citací vypočítaný z dat získaných ze Scopus API,
- WosHIndexSelfExcluded - h-Index bez vlastních citací vypočítaný z dat získaných z Web of Science,
- ScopusHIndexSelfExcluded - h-Index bez vlastních citací vypočítaný z dat získaných ze Scopus API.

- WosHIndexAllExcluded - h-Index bez vlastních citací všech autorů vypočítaný z dat získaných z Web of Science,
- ScopusHIndexAllExcluded - h-Index bez vlastních citací všech autorů vypočítaný z dat získaných ze Scopus API,
- JobTitleId - identifikátor přiřazeného pracovního zařazení,
- GroupId - identifikátor přiřazené skupiny,
- EndYear - rok odchodu autora z univerzity,
- StartYear - rok příchodu autora na univerzitu,
- Login - login autora přidělený VŠB-TUO.

Group
Tabulka Group uchovává skupiny, jako jsou fakulty, katedry nebo odborné skupiny.

- Id,
- Name - Název skupiny,
- Code - kódové označení skupiny slouží zejména pro katedry (nemusí být vyplněno)
- Level - úroveň, na které se skupina v hierarchii nachází.

Publications
Tabulka k uchování informací o publikacích.

- Id
- Name - název publikace,
- Published - rok publikování,
- WosId - id přidělené publikaci v IS Web of Science,
- ScopusId - id přidělené publikaci v IS Scopus, jeden z atributů WosId nebo ScopusId pro jednu entitu musí nabývat hodnoty různé od **NULL**,
- CitationCount - počet citací včetně vlastních citací,
- CitationCountAllAuthorsExcluded - počet citací bez vlastních citací všech autorů.

AuthorPublications
Vazební tabulka mezi tabulkami Author a Publications.

- AuthorId - id autora odkazující do tabulky *Author*,
- PublicationId - id publikace odkazující do tabulky *Publiactions*,
- CitationCountSelfExcluded - Počet citací bez vlastních citací.

JobTitle
Tabulka k uchování informací o pracovních zařazeních.

- Id,
- Name - název pracovní pozice.

6.4 Použité technologie

Při vytváření výsledné aplikace jsem se setkal s mnohými technologiemi. Skoro všechny použité technologie jsem již znal ze školy, případně z praxe. Přesto v této kapitole popíši několik použitých technologií.

6.4.1 C#

C# je programovací jazyk vyvinutý firmou Microsoft [15]. První verze jazyka byla vydána v roce 2002 a o rok později byl jazyk standardizován pod označení standardu ISO/IEC 23270. Ke dni odevzdání této diplomové práce je poslední verzí jazyka C# verze 6, která byla vydána v roce 2015. C# je typově bezpečný, objektově orientovaný a garbage kolektovaný programovací jazyk, který umožňuje vývojářům vytvářet bezpečné a robustní aplikace, které běží na .NET frameworku. Vývojáři jsou schopni za pomocí jazyka C# vytvářet Windows aplikace, klient-server aplikace, databázové aplikace a mnoho dalších. Visual C# poskytuje velmi propracovaný editor zdrojového kódu, přehledné uživatelské rozhraní a integrovaný nástroj pro ladění zdrojového kódu. Syntaxe jazyka C# je přehledná a používá se zde složených závorek pro uzavření kódu do bloků, stejně, jako je tomu u jazyků C nebo C++. C# dovoluje vytvářet nullable datové typy, lambda výrazy, generické datové typy a metody i využití LINQ výrazů.

6.4.2 ASP.NET Web Forms

ASP.NET Web Forms je sada technologií v .NET frameworku pro vytváření webových aplikací a XML webových služeb vyvinutá ve firmě Microsoft [15]. Stránky ASP.NET jsou spuštěny na serveru a generují kód značkovacího jazyka jako je například HTML, který je následně zaslán prohlížeči počítače nebo mobilního zařízení na straně klienta. Stránky ASP.NET používají zkompilovaný, událostmi řízený programovací model, který umožňuje oddělení aplikační logiky

od uživatelského rozhraní. ASP.NET stránky a XML webové služby obsahují logiku na straně serveru, jež je psána v některém z programovacích jazyků kompatibilních s .NET frameworkem firmy Microsoft, jako je například jazyk Visual C# .NET.

6.4.3 SQL Server

SQL Server se řadí mezi systémy řízení báze dat, byl vyvinut firmou Microsoft [15] a je určený pro správu relačních databází. Komunikace s SQL serverem probíhá pomocí standardizovaného SQL jazyka nebo procedurálního rozšíření SQL Serveru Transact-SQL. Samotný SQL Server se skládá z několika částí - Database Engine, Integration Services, Analysis Services, Reporting Services a dalších.

V diplomové práci jsem využil pouze základní služby SQL Serveru a to Databse Engine. Database Engine je základní služba pro ukládání, zpracovávání a zabezpečení dat. Database Engine poskytuje řízený přístup a rychlé zpracování transakcí, aby splňoval požadavky i těch nejnáročnějších aplikací spotřebovávajících data. Database Engine se používá k vytvoření relační databáze pro zpracování transakcí nebo pro analytické zpracování dat online. To zahrnuje vytváření tabulek pro uložení dat a databázové objekty, jako jsou indexy, pohledy nebo uložené procedury pro prohlížení, správu a zabezpečení dat.

6.4.4 XML

XML je jednoduchý velmi flexibilní textový formát odvozený z SGML, který byl vytvořen komunitou W3C [16]. Ta také udržuje jeho specifikaci. Původně navržený, aby řešil elektronické publikování ve velkém měřítku. XML také hraje velkou roli při výměně nejrůznějších dat nejen na webu.

XML popisuje třídu datových objektů uložených v počítačích a částečně popisuje chování programů, které zpracovávají tyto objekty. Takové objekty se nazývají XML dokumenty. XML dokumenty jsou tvořeny z jednotek nazvaných entity, které obsahují text nebo binární data. Text se skládá ze znaků, z nichž některé tvoří data v dokumentu a jiné tvoří samotné značkování. Značkování kóduje popis rozložení uložených dokumentů, strukturu a libovolné atributy formované v párech atribut-hodnota. Ukázka XML dokumentu je uvedena v příloze B.

6.4.5 HTML

HTML je jednou ze základních technologií pro vytváření webových stránek udržovanou komunitou W3C [16]. HTML je jazyk, který popisuje strukturu stránek použitím značkování. Obsah dokumentu je možné rozdělit na části pomocí značkovacích tagů, které potom dohromady s obsahem tvoří jednotlivé prvky dokumentu. HTML umožňuje publikovat online dokumenty s nadpisy, texty, fotkami, tabulkami, apod. Umožňuje načtení online informací, prostřednictvím hypertextových odkazů. Tvůrce dokumentu může navrhovat formuláře pro provádění transakcí

se vzdálenými službami, pro objednávání zboží nebo vytváření rezervací, ale také může přímo v dokumentu použít přehrávač hudby a videa nebo jiné aplikace.

6.4.6 CSS

CSS je jazyk, který slouží pro prezentaci webových stránek [16]. Umožňuje autorovi stránek nastavit barvy, písma a rozvržení stránky. CSS také dovoluje přizpůsobit prezentaci stránky různým typů zařízení jako jsou velké obrazovky nebo menší obrazovky mobilních zařízení a tabletů. CSS není závislý na HTML. Může být použit s jakýmkoliv značkovacím jazykem založeným na XML. Oddělení struktury od prezentace (CSS od HTML) poskytuje snadnější údržbu stránek, sdílení stylů mezi stránkami, ale i možnost přechodu stránek na jiné prostředí. Při oddělení prezentace je potřeba při psaní stylů využití CSS selektorů pro výběr prvků stránky, kterým mají být styly přiřazeny.

6.4.7 JavaScript a JQuery

JavaScript je skriptovací jazyk s původním názvem ECMAscript pojmenovaným po firmě Ecma, která jej vyvinula. Skript jazyka JavaScript není před spuštěním, stejně jako ostatní skriptovací jazyky, předkompilovaný. V případě webového prohlížeče je skript spuštěn obvykle po stažení stránky nebo po akci uživatele na stránce. Skriptování dělá stránky dynamičtějšími a taky poskytuje možnost upravit stránku nebo poslat obsah na stránku nebo ze stránky na server bez znovunačtení stránky. World Wide Web Consortium [16] také vyvinulo rozhraní DOM, které ulehčuje práci díky dynamickému přístupu a aktualizaci obsahu, struktury a stylu dokumentu.

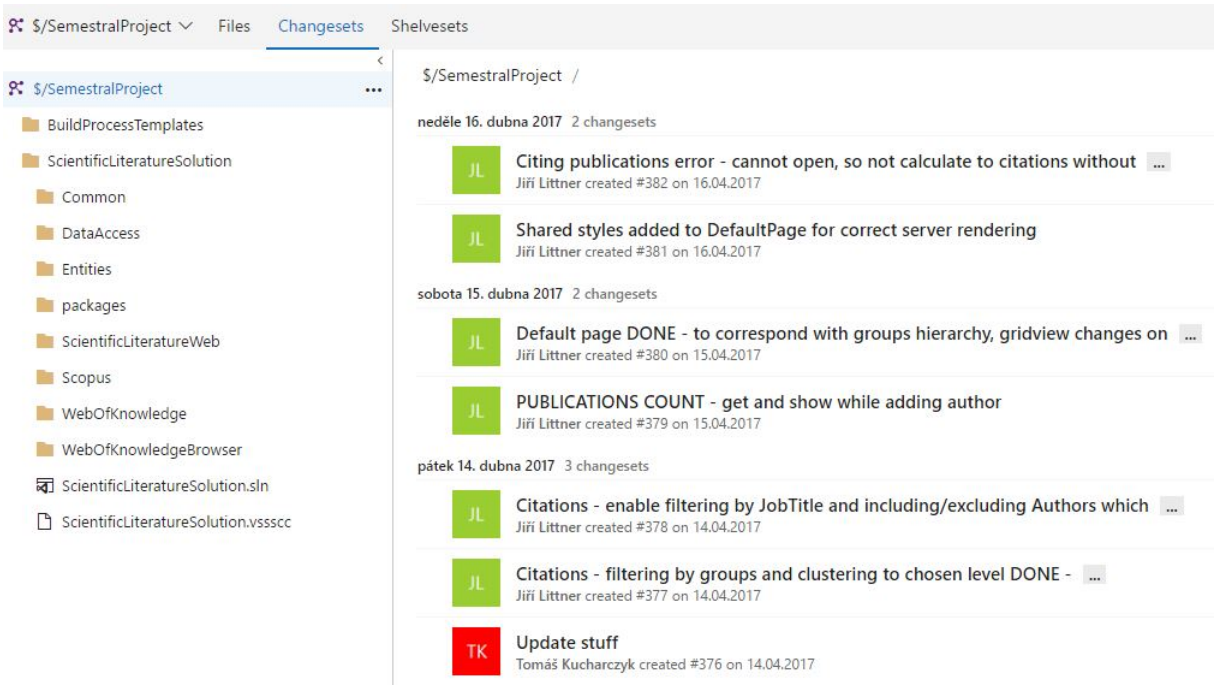
Jquery je JavaScriptová knihovna, která dovoluje snadnější a rychlejší práci s dokumentem jako např. zpracovávání událostí, animace, manipulace s prvky stránky a další. Jedná se dnes o velice rozšířenou knihovnu pro práci s HTML dokumenty.

6.4.8 Team Foundation Server

Team Foundation Server je nástroj pro verzování zdrojového kódu. Umožňuje vývoj software týmech, poskytuje sadu nástrojů pro spolupráci při vývoji, které jsou integrovány s editory zdrojových kódů. Zdrojový kód je udržován v centrální repository, kde by měl být kód udržován ve funkční verzi. Každý vývojář má repository stáhnutou lokálně, ve které pracuje a po dokončení implementace nějaké části výsledného systému je zdrojový kód synchronizován s centrální verzí a takto mezi sebou vývojáři sdílejí změny v kódu. TFS také nabízí spoustu dalších nástrojů pro agilní metodiky vývoje software jako jsou Scrum nebo Kanban, umožňuje tak pomocí dashboardů kontrolovat postup vývoje, jeho rychlost a definovat iterace vývoje výsledného produktu. TFS tak nabízí možnost pravidelného sestavování zdrojových kódů se spouštěním automatizovaných testů, čímž napomáhá včasnému odhalení chyb a tím k údržbě zdrojového kódu. S kolegou jsme pro vývoj využili verze TFS[2], kterou zdarma poskytuje firma Microsoft. Na obrázku 11 jsou

[2]Námi používané TFS dostupné na URL adrese `https://tomasjirka.visualstudio.com/SemestralProject`

vidět některé změny, které jsme pravidelně po dokončení části implementace s kolegou nahrávali na TFS server.



Obrázek 11: Ukázka změn v TFS při vývoji

7 Popis implementace

Vývoj celého systému probíhal volnějším iteračním způsobem založeným na agilních vývojových metodikách. To znamená, že jsme měli s vedoucím práce pravidelné konzultace, na kterých jsem vždy ukázal postup ve vývoji a domluvili jsme se na dalších úkolech, které jsem měl zpracovat. Musel jsem tak reagovat na funkční požadavky vedoucího práce a následně je reflektovat ve vyvíjeném systému. Pro vývoj systému jsme s kolegou vybrali technologii ASP.NET WebForms 6.4.2 s použitím programovacího jazyka C# 6.4.1. Pro sdílení a správu zdrojového kódu jsme s kolegou zvolili verzovací nástroj TFS 6.4.8 od firmy Microsoft kvůli předešlým pozitivním zkušenostem. Vyvíjený systém jsme také nasazovali na server **dbedu.cs.vsb.cz**, který nám vedoucí poskytl, aby mohl sám doplnit OrcId a ResearcherId autorů z Web of Science do naší databáze. Tím umožnil vyhledávání ve Web of Science podle Id autorů a né podle jména, což vedlo k upřesnění výsledků, protože tak bylo zmenšeno riziko zobrazení publikací různých autorů způsobené shodou jmen.

V kapitole jsou popsány implementace přístupových objektů k API a webu, způsob plnění vlastní databáze získanými daty, ORM nezbytné pro práci s daty v aplikaci a v poslední podkapitole je popsána klíčová funkcionalita samotné webové aplikace.

7.1 Popis implementace přístupového objektu ke Scopus API

Před začátkem vývoje jsem si vytvořil testovací aplikaci, abych vyzkoušel práci se Scopus API. Jak jsem již uvedl, Scopus nabízí celkem 11 API, ze kterých se dají získat informace, proto jsem v testovací aplikaci nejprve vytvářel požadavky na různá z nich, a studoval jsem struktury navrácených odpovědí.

Při vytváření HTTP požadavku na API jsem do URL adresy uchované globálně doplnil ID autorů, pro která se vracel seznam publikací. Všechna API, se kterými jsem pracoval, jsou připravena na stránkování a omezena tak na velikost stránky maximálně 50 autorů. Proto bylo nezbytné sestavovat URL adresy s id autorů i s filtry dynamicky. Do hlavičky HTTP požadavku je následně přidán klíč k API pro autentizaci, je zde určena metoda požadavku *GET* a také návratový typ, který jsem si zvolil **application/xml**.

Kvůli maximálnímu počtu parametrů v URL adrese pro stránkování jsem musel stránkování seznamu autorů simulovat. Vypočítá se počet stránek a v každé iteraci cyklu se dynamicky sestaví adresa a odešle se požadavek na server. Příklad odesílané URL adresy je ve výpisu zdrojového kódu 3.

```
"http://api.elsevier.com:80/content/author?field=dc:identifier,document-count,
    cited-by-count,citation-count,given-name,surname,h-index&author_id
    =23392363600,34977261800,24768171500,24729504200,23392385900"
```

Výpis 3: URL požadavku na server

Při implementaci požadavků jsem se potýkal s nepříjemnou situací. Při získávání citací bez vlastních citací z Citations Overview API jsem podle dokumentace k API [14] přidal jako parametr do URL *citation* s hodnotou *exclude-self*. API následně dle dokumentace mělo vrátit počet citací publikace bez autocitací, ale není zde již konkrétně napsáno, jestli se jedná o citace bez vlastních citací daného autora nebo o citace bez autocitací všech autorů. Mýlil jsem se, když jsem počítal, že mi tak API vrátí počet citací bez vlastních citací autora, protože API takto vrací počet citací bez vlastních citací všech autorů. Abych získal počet citací bez autocitací daného autora, musel jsem autorovo ScopusId předat jako parametr *author_id*, pak API vrátí požadovaný výsledek. API vrací seznam publikací s citacemi a pro každou publikaci také seznam jejich autorů. Ukázka XML 6.4.4 dokumentu vráceného z Citations Overview API je v příloze B.

Z XML dokumentu jsem potřeboval data získat a uchovat v objektech, proto jsem vytvořil nový projekt se sadou entit odpovídající položkám dokumentu navráceného z API serveru.

Pro parsování odpovědí jsem využil integrované .NET knihovny LINQ to XML, se kterou jsem do té doby neměl zkušenosti, ovšem práce s ní byla skoro stejná jako s knihovnou LINQ to Objects, jež slouží pro práci s objekty. Protože jsem zvyklý pracovat s LINQ to Objects jako s řetězením metod a ne jako s dotazovací syntaxí, použil jsem stejný způsob parsování i pro XML, což je vidět v ukázce 4.

```
List<Author> authors = elements.Select(x => new Author
{
      ScopusId = x.Descendants(dc + "identifier").First().Value.Split(':')[1],
      GivenName = x.Descendants("given-name").First().Value,
      Surname = x.Descendants("surname").First().Value
}).ToList();
```

Výpis 4: Rozebrání odpovědi serveru pomocí LINQ to XML

Při implementaci objektu pro parsování odpovědí jsem se musel rozhodnout, v jaké struktuře budou publikace uchovávány. První možností bylo uchovávat publikace jako seznam pro každého autora. Toto řešení mi však nepřišlo jako optimální, protože jsem v dalším kroku potřeboval pro každou publikaci odeslat dva požadavky na Citations Overview API pro informace o citacích publikací a odpovědi zpracovat. Pro kolekci autorů, která by opět obsahovala autory, kteří spolupracovali na některých publikacích, by to znamenalo získávat citační informace o těchto společných publikacích vícekrát a při vkládání do databáze by se publikace vložila stejně jen jednou. Proto jsem strukturu uchovávání publikací obrátil a získané publikace všech autorů ukládal do jedné kolekce, kde každá publikace obsahuje informace o svých všech autorech.

V kolekci publikací se v danou chvíli opět vyskytují společné publikace autorů vícekrát, proto jsem musel duplicity z kolekce odstranit. Protože publikace již obsahovaly seznam autorů, potřeboval jsem přenést citace bez vlastních citací autora z publikace k odstranění do publikace se stejným ScopusId, která v kolekci zůstane. Takto jsem vytvořil seznam publikací bez duplicit, kde publikace obsahují informace o všech autorech a jejich citacích bez vlastních citací autorů.

Při získávání publikací ze Scopus Search API se občas stalo, že v navráceném XML dokumentu odpovědi byla u některých publikací uvedena všechna jména autorů dvakrát, proto jsem vytvořil metodu pro procházení autorů v publikacích a smazání duplicit podobně jako u publikací.

Ukázka 5 zobrazuje způsob odstranění duplicitních publikací.

```
for (int i = 0; i < totalPublicationsList.Count - 1; i++)
{
   for (int j = i + 1; j < totalPublicationsList.Count; j++)
   {
      if (totalPublicationsList[i].ScopusId ==
            totalPublicationsList[j].ScopusId)
      {
         foreach (var authorJ in totalPublicationsList[j].Authors)
         {
               var authorI = totalPublicationsList[i].Authors
                     .FirstOrDefault(x => x.ScopusId == authorJ.ScopusId);
               if(authorJ.CitationCountExcludedSelfOnly >
                  authorI.CitationCountExcludedSelfOnly)
               {
                  authorI.CitationCountExcludedSelfOnly = authorJ.
                      CitationCountExcludedSelfOnly;
               }
         }
         totalPublicationsList.Remove(totalPublicationsList[j]);
      }
   }
}
```

Výpis 5: Odstranění duplicitních publikací ze seznamu

7.2 Popis implementace přístupového objektu k Web of Science

Jak jsem již uvedl v kapitole 4, data z Web of Science jsou získávána přímo z webové aplikace.

V původní verzi aplikace byla data získávána pomocí komponenty webového prohlížeče WebBrowser, která je součástí knihovny WinForms frameworku .NET. Testování implementace bylo velice časově náročné. Při testování jsem 11 hodin čekal na výsledek a aplikace spadla. Při bližším přezkoumání jsem zjistil, že komponenta WebBrowser neuvolňuje paměť a došlo tak k vyčerpání paměti.

Proto jsem hledal alternativu k této komponentě a narazil jsem na 3 open source řešení, kterými jsou GeckoFX (obal jádra používaného v prohlížeči Firefox), Awesomium a CefSharp,

což jsou obaly jádra Chromium použitého v prohlížeči Google Chrome. Všechny tři nástroje hospodárněji pracují s pamětí oproti komponentě WebBrowser. Problémem s těmito komponentami je, že jejich instance musí být vytvořeny pouze jednou v daném procesu, proto jsem ustoupil i od těchto řešení a začal jsem implementovat získávání dat z Web of Science pomocí HTTP požadavků.

Mým prvním úkolem bylo umožnit získávání klíče **SID**. Vytvořit HTTP požadavek pro získání klíče **SID** nebyl žádný problém. Framework .NET poskytuje několik tříd pro vytváření a zpracování HTTP požadavků, já jsem využil konkrétně třídy HttpWebRequest a HttpWebResponse . Tento požadavek má nastavenu metodu *GET* a vždy jako první směřuje na domovskou URL adresu úvodní vyhledávací stránky Web of Science, jež je uvedena v [2]. Server následně v obsahu odpovědi vrátí HTML dokument stránky, kde je ve skrytém poli uloženo **SID**.

Když jsem již získal aktivní **SID**, potřeboval jsem získat stránku se seznamem publikací pro vybraného autora. Pokoušel jsem se vytvořit požadavek s metodou *POST*, který by mi HTML dokument se seznamem publikací vrátil, ale nedařilo se mi jej vytvořit. Proto jsem si nastudoval odesílané HTTP požadavky v prohlížeči, abych věděl, jaké požadavky a s jakým obsahem jsou odesílány na server. V těchto požadavcích je uvedena i URL adresa, na kterou byly požadavky odeslány, ale adresa lze vyčíst i z HTML dokumentu navráceným serverem. Dokument obsahuje formulář s atributem *action*, který určuje akci na serveru, která se má po potvrzení formuláře zavolat. Toho lze docílit i odesláním HTTP požadavku na adresu, která se skládá z URL adresy právě navštívené stránky, ke které je připojena akce z formuláře. Při přidání obsahu do požadavku pak akce provede svůj úkol a server vrátí odpověď.

Protože formulář pro vyhledání publikací autora obsahuje mnoho polí pro odeslání na server, pak také obsah HTTP požadavku je poměrně rozsáhlý. Když jsem vytvořil funkcionalitu pro odeslání požadavku s metodou *POST* a přidal jsem do obsahu požadavku stejná data, jaká odesílá prohlížeč, IS Web of Science vracel HTML dokument s chybovou hláškou. Po mnoha pokusech a úpravách požadavku jsem přidal jako obsah požadavku pouze základní informace a server nakonec odeslal v odpovědi HTML dokument s publikacemi mnou požadovaného autora. Pro vyhledání publikací autora je nejlepší řešení použít jeho *OrcId* nebo *ResearcherId*, abychom získali nejpřesnější výsledek. Zdaleka ne všichni autoři mají u svého profilu ve Web of Science vyplněny identifikátory, proto nemohou být vyplněny ani ve vlastní databázi aplikace. Aplikace tak při vyhledávání publikací autora využívá podmínek, ve kterých se kontroluje existence ID autorů. Kdy existuje alespoň jeden, vyhledávají se publikace podle něj, v opačném případě jsou publikace vyhledány podle jména autora, ale pro zpřesnění výsledků je připojen druhý vyhledávací výraz logickou spojkou **AND**, kde je uvedena organizace. Ukázka 6 vyobrazuje obsah HTTP požadavku *POST* na server s vyhledáním podle jména a organizace autora.

```
"fieldCount=1&action=search&product=WOS&search_mode=GeneralSearch&value%28
    input1%29=Voznak+Miroslav&value%28select1%29=AU&value%28bool_1_2%29=AND&
    value%28input2%29=Technical+University+Ostrava"
```

Výpis 6: Ukázka obsahu HTTP požadavku

Protože jsem potřeboval získávat ID autorů, aby mohla aplikace vypočítat počet citací bez autocitací všech autorů, rozhodl jsem se pro procházení již otevřenými publikacemi. Po vyhledání autora aplikace rozklikne první publikaci a rovnou tak přejde na její detailní stránku, ta umožňuje přejít odkazem na další, což je vidět na obrázku 5. Stejně tak zde lze přejít na citující publikace. Přechody mezi stránkami jsou uskutečněny pomocí HTTP požadavků s metodou *GET* na adresy získané z odkazů v navráceném HTML dokumentu.

Průchod citujícími publikacemi se provádí naprosto stejně jako průchod samotných publikací autora, a tudíž zde volám nepřímou rekurzí stejné metody s počtem zanoření rovno 1.

Při získávání citujících publikací skrze odkaz s počtem citací publikace jsem narazil na problém, který jsem již popsal v kapitole 4. Pro publikace, které neumožňují zobrazit své citující publikace není možné zjistit, zdali se jedná o citace bez autocitací nebo ne, proto aplikace počítá s horším případem, že se jedná o autocitace, a proto se do citací bez autocitací nezapočítávají.

Pro parsování HTML dokumentu navráceného serverem je použit Html Agility Pack [17] dostupný z Nuget správce balíků. Html Agility Pack je agilní HTML parser, který vytváří DOM pro čtení nebo zapisování a podporuje XPATH nebo XSLT. Jedná se o .NET knihovnu tříd, která umožňuje parsovat HTML soubory. Parser je tolerantní vůči chybně sestavenému HTML dokumentu. Objektový model je velmi podobný tomu, co poskytuje knihovna System.Xml, ale pro HTML dokumenty.

Já jsem pro získání dat z dokumentu využil možnosti XPATH. Následující ukázka 7 zobrazuje získání odkazu na citující publikace, z jehož atributu *href* se dále získává URL adresa.

```
"(//div[@class='l-column-sidebar2']/div[@class='block-text'])[1]//a"
```

Výpis 7: Ukázka XPATH pro získání odkazu z dokumentu

V původní verzi implementace se po dokončení procházení publikací daného autora ukládaly publikace do jedné kolekce a citující publikace těchto publikací do druhé. Pro více autorů se tak získávaly jejich společné publikace vícekrát, podle počtu spoluautorů publikace, pro které se publikace získávají. Před vložením do databáze musely být duplicitní publikace stejně odstraněny, proto jsem tento způsob uchovávání dat optimalizoval.

Optimalizace implementace spočívala v kontrole existence již získaných publikací ještě před spuštěním algoritmu pro získávání citujících publikací právě procházené publikace. Pro tuto optimalizaci jsem zvolil nahrazení seznamů s publikacemi List<Publication> slovníkem Dictionary

<string, Publication>, kde klíčem je ID publikace a hodnotou je objekt samotné publikace, která nese informace o svých autorech a citujících publikacích.

Při kontrole existence publikace tak sice musí být procházen pro každou publikaci vždy již uložený seznam publikací, který postupně narůstá, ale v porovnání s ušetřenou dobou potřebnou pro získávání dat je doba průchodu slovníku mnohonásobně menší, což je vidět v naměřených výsledcích v kapitole 9.

Po získání dat všech autorů, je důležité vypočítat počet citací bez autocitací a počet citací bez autocitací všech autorů. Obecný algoritmus pro výpočet obou ukazatelů je již uveden v kapitole 5. V případě počtu citací bez autocitací autora jen doplním, že určení existence citující publikace v seznamu publikací autora je prováděno na základě *Accession number* publikace, které je jednoznačné pro každou publikaci, proto jsou výsledky výpočtů přesné.

V případě výpočtu citací bez autocitací všech autorů je vhodné připomenout, že ne všichni autoři mají ve Web of Science vyplněno své OrcId a ResearcherId. Proto jsou při získávání citačních informací získávány kromě ID autorů i jejich jména. Při následném výpočtu citací je pak existence autorů porovnávána nejprve podle OrcId, když nebylo vyplněno, tak podle ResearcherId a v případě obou chybějících ID jsou autoři porovnáváni podle jmen. Tyto 3 případy kontroly umožňují minimalizovat výskyt chyb ve výsledku výpočtu citací.

7.2.1 Plnění databáze daty

Vedoucí diplomové práce mi poskytl seznam autorů Fakulty elektrotechniky a informatiky se zařazením do kateder, který jsem měl přidat do databáze. Seznam autorů byl v souboru formátu PDF, ze kterého bych data aplikací jednoduše nepřečetl, proto jsem si jej převedl do jiného formátu a potřebná data uložil do souboru CSV. V době zpracovávání souboru s autory jsem již věděl, že budu v IS Web of Science dočasně vyhledávat autory podle jména, a protože je pro vyhledání všech publikací autora lepší použít jméno bez diakritiky, musel jsem před vložením autorů do databáze jména upravit.

V aplikaci jsem dočasně vytvořil metody pro načtení souboru, úpravu jmen a vygenerování SQL skriptu. Metoda pro úpravu jmen obsahuje dva textové řetězce, v prvním je sada znaků s diakritikou a ve druhém je sada odpovídajících znaků bez diakritiky. Pozice odpovídajících si znaků v řetězcích jsou si rovny, proto je pak možné jednoduše procházet jméno autora po jednotlivých znacích a konkrétní znak porovnávat se znaky v prvním řetězci a v případě shody jej nahradit znakem z druhého řetězce na stejné pozici. Po úpravě jmen jsem si nechal vygenerovat SQL skript pro vložení autorů do databáze.

Pro autory ze seznamu jsem potřeboval uchovat i jejich Id z IS Scopus pro získávání dat ze Scopus API. Mým dalším úkolem bylo projít ručně všechny autory ze seznamu a vyhledat v IS Scopus jejich Id. Pro tato id jsem si v další metodě taky nechal vygenerovat SQL skript, tentokrát ale pro aktualizaci již existujících dat. Usnadněním práce by bylo získat Id autorů programově podle jmen z API, ale toto API vrací více výsledků v případě shody jmen stejně,

jako to dělá IS Scopus a aplikace by neměla jak rozeznat, který z autorů je správný.

Plnění databáze získanými daty z API provádí samotná webová aplikace tak, že nejprve získá data z API nebo webu pomocí již uložených id autorů a následně data uloží do databáze. Data jsou získávána a ukládána do databáze postupně. Před dokončením získání všech dat jsou data uložena do pracovních tabulek (viz podkapitolu **Vlastní databáze** kapitoly 6) a až po dokončení aktualizace jsou data překopírována do hlavních tabulek.

Jako první jsou získávána data ze Scopus API. Získávání dat probíhá v dávkách podle skupin autorů (resp. podle kateder). Po získání všech dat pro autory jedné katedry jsou tato data uložena do pracovních tabulek a pokračuje získávání dat pro další katedru.

Když jsou získána data všech autorů ze Scopus API, pokračuje aplikace se získáváním dat z Web of Science. Data jsou z Web of Science získávána ve dvou fázích. V první fázi jsou získávána data o publikacích samotných opět podle kateder, o čemž píše více ve své práci [3] kolega, ve druhé fázi jsou již získávány počty citací jednotlivých publikací.

Rozdíl oproti kolegově způsobu plnění dat do databáze je, že počty citací nemohou být uloženy do pracovních tabulek po získání dat každé katedry, ale až po získání všech dat. Důvodem je výpočet citací bez autocitací všech autorů, který se provádí právě po získání všech dat a počítá se s autory uvedenými u citujících publikací, kteří v databázi nejsou, o čemž jsem již psal v kapitole 5. Po získání všech dat jsou vypočteny i počty citací bez autocitací autora a po obou výpočtech jsou všechna data uložena do pracovních tabulek.

Protože jsou h-indexy uloženy v tabulce společně s informacemi o autorech, pak jsou tyto h-indexy vypočítávány až po překopírování dat z pracovních tabulek do hlavních tabulek pro zachování konzistence dat. Přepočtení h-indexů probíhá také ve dvou krocích. Nejprve jsou z databáze získány všechny počty citací publikací pro Scopus, publikace jsou rozděleny dle autorů a následně jsou vypočítány 3 h-indexy. Jeden je vypočítán z počtu všech citací pro autora, druhý z počtu citací bez autocitací autora a třetí z počtu citací bez autocitací všech autorů. Následně jsou h-indexy uloženy do databáze. Stejný proces probíhá i pro výpočet h-indexů pro počty citací z Web of Science a databáze tak uchovává celkem 6 ukazatelů h-index pro každého autora. V kapitole 5 jsem již algoritmus pro výpočet h-indexu popsal.

7.2.2 ORM

Abych mohl pracovat s daty ve webové aplikaci a ukládat získaná data do databáze, potřeboval jsem vytvořit ORM. Uvažoval jsem nad použitím nějakého nástroje pro jeho tvorbu, ale nakonec jsme se s kolegou shodli na vlastním ORM. Do řešení jsem přidal nový projekt *DataAccess*, který představuje onu mapovací vrstvu z relační databáze do objektů v aplikaci. Pro každou z tabulek databáze jsem vytvořil třídu a v ní naimplementoval základní CRUD operace. Projekt obsahuje i třídu *Database* s připojovacím řetězcem, metodami pro připojení a odpojení z databáze, vytvoření SQL příkazu, metodami pro správu transakcí a metodami pro spuštění CRUD operací. Metody této třídy využívají všechny ostatní třídy v projektu. S postupným vývojem se všechny třídy v tomto projektu rozrůstaly o nové metody. Největší změny se v projektu udály až při dalších úpravách stránky pro skupinu autorů, o čemž píši více i s ukázkou SQL dotazů v následující podkapitole.

7.3 Popis implementace webové aplikace

Pro stylování výsledných formulářů jsem použil framework Bootstrap pro vytváření přehledného uživatelského rozhraní, díky jehož použití se stává aplikace též responzivní.

Ve všech formulářích kromě administrační části bylo nezbytné umožnit uživateli vybrat skupinu pro zobrazení autorů. Aby se neopakovala funkcionalita pro výběr skupiny autorů napříč aplikací, vytvořil jsem společný ovládací prvek, který obsahuje seznam skupin a je použit ve formulářích.

Skupiny jsou vytvořeny ve stromové struktuře, proto jsem zvolil pro zobrazení uživateli ovládací prvek TreeView. Prvek TreeView je plněn položkami TreeNode, které obsahují odkaz na své potomky. Po vybrání skupin z databáze jsou skupiny uchovány v seznamu, kde každá skupina má odkaz na svou rodičovskou skupinu. Abych mohl vytvořit stromovou strukturu položek TreeNode, potřeboval jsem využít algoritmu pro průchod stromovou strukturou, který jsem uvedl v kapitole 5. Abych však mohl použít tento algoritmus pro zařazení skupin, bylo potřebné vždy ze seznamu skupin získat skupiny procházené úrovně a následně z nich získat skupiny s rodičovskou skupinou stejnou jako je předána v parametru metody. Ukázka zařazení skupin do hierarchie je zobrazena v ukázce 8

```
private void AssignGroupsToHierarchy(TreeNode treeNode, int parentGroupId, int
    level)
{
   var groupsToAssign = GroupsToTreeview.Where(x => x.Level == level).ToList();
   if (groupsToAssign != null && groupsToAssign.Any())
   {
      var approptiateDescendantGroups = groupsToAssign.Where(x => x.
          ParentGroupId == parentGroupId).ToList();
      if (approptiateDescendantGroups != null && approptiateDescendantGroups.
          Any())
      {
         foreach (var appropriateDescendantGroup in approptiateDescendantGroups)
         {
            var groupName = appropriateDescendantGroup.Code.HasValue ?
                appropriateDescendantGroup.Code.Value + " - " +
                appropriateDescendantGroup.Name : appropriateDescendantGroup.
                Name;
            groupName = ShowLevels ? groupName + " (" +
                appropriateDescendantGroup.Level + ")" : groupName;
            var treeNodeToAssign = new TreeNode(groupName,
                appropriateDescendantGroup.Id.ToString());
            treeNode.ChildNodes.Add(treeNodeToAssign);
            AssignGroupsToHierarchy(treeNodeToAssign, appropriateDescendantGroup
                .Id, level + 1);
         }
      }
   }
}
```

Výpis 8: Ukázka zařazení skupin do stromové struktury

Při získávání autorů z databáze pro vybranou skupinu jsou získáni všichni autoři vybraného podstromu skupin. Pro konstrukci SQL dotazu na celý podstrom jsem tak potřeboval z podstromu skupin získat všechny jejich ID. Protože vybraná položka v TreeView obsahuje informaci o celém podstromu, využil jsem základní verzi preorder algoritmu průchodu podstromu uvedeného v kapitole 5 a rozšířil jsem jej pouze o získávání ID skupin.

Sestavování výsledného SQL dotazu, konkrétně klauzule **WHERE** pak probíhá dynamicky podle počtu ID skupin. Stejně tak jsou pro dotaz generovány jeho parametry. Tímto dynamickým způsobem je umožněno administrátorovi aplikace libovolně přidávat skupiny do stromové

struktury a přiřazovat k nim autory, aniž by byla aplikace nějak omezena v zobrazování autorů. Výsledný dynamicky sestavený SQL dotaz je uveden v příloze C

Seznam autorů ve formuláři umožňuje uživateli také seřazovat autory dle libovolného atributu. V implementaci požadavku **4** kapitoly 3, se tak ve formuláři s přehledem provádí seřazování autorů přímo nad daty v databázi. V ORM jsem vytvořil metodu pro dynamické sestavení klauzule **ORDER BY** podle předaného výrazu a směru seřazení, která je následně přidána k SQL dotazu.

Jednou z funkcí požadavku **5** v kapitole 3 bylo zobrazení získaných dat v tabulkách ve formuláři citací autorů. Při každém zobrazení formuláře, se musí před načtením dat do tabulek formuláře sečíst počty citací publikací autora. Sčítání těchto publikací probíhá již v databázi při výběru autorů, kde je k autorům také připojena skupina, ke které jsou přiřazeni. Uživatelem zvolené filtry ve formuláři jsou prováděny také na databázové úrovni opět dynamickou konstrukcí klauzule **WHERE**, což je vidět v ukázce, která je uvedena v příloze C. Jelikož není jisté, který filtrační výraz uživatel zvolí, musela již být klauzule vytvořena s nějakým výrazem, který však neovlivní výsledky získaných dat i bez filtračních výrazů. Pro tento účel jsem zvolil výraz **1=1**, což je tautologie a tudíž jeho pravdivostní hodnota je vždy *pravda* a nijak neovlivní výsledky získaných dat z databáze. Výsledný SQL dotaz v ukázce C obsahuje dva podobné poddotazy, ale mají různé výrazy v klauzuli **WHERE**. Zde jsou agregovány počty citací pro autora. Jak jsem již zmiňoval v podkapitole **Vlastní databáze** kapitoly 4, tabulka *Publications* slouží pro uchování dat z obou citačních databází, proto jsou zde použity 2 poddotazy a jen výraz ve **WHERE** rozlišuje, jestli mají být získána data pro Scopus nebo pro Web of Science.

Výsledkem SQL dotazu byl seznam autorů s jim přiřazenými skupinami. S tímto seznamem by bylo obtížné pracovat dále v aplikaci, proto jsem v ORM vytvořil metodu pro obrácení struktury, kde nejsou zařazeny skupiny k autorům, ale autoři ke skupinám. Pro seskupení autorů do skupin jsem využil metody *GroupBy()* z LINQ to Objects. Abych mohl tuto metodu využít, nejprve jsem převedl data získaná z databáze do kolekce typu dynamic a poté jsem mohl metodu využít pro rozdělení dat do objektů. Práce se získanými daty je vidět v ukázce zdrojového kódu 9.

```
var dynamicList = new List<dynamic>();
while (reader.Read())
{
   dynamic dynamicObject = new ExpandoObject();
   dynamicObject.GroupId = reader.GetInt32(0);
   //...
   dynamicList.Add(dynamicObject);
}
var groups = new List<Group>();
if (dynamicList.Any())
{
   groups = dynamicList.GroupBy(x => x.GroupId, x => x, (key, group) =>
               new Group()
   {
      Id = key,
      Code = group.First().GroupCode,
      Name = group.First().GroupName,
      Authors = group.Select(g => new Author
      {
            Id = g.AuthorId,
            //...
            }).ToList()
   }).ToList();
}
```

Výpis 9: Ukázka seskupení autorů do kateder

Aplikace umožňuje zobrazit také uživateli tabulku s agregovanými hodnotami skupin. Zde si uživatel může zvolit také do jaké úrovně podstromu vybrané skupiny chce hodnoty rozdělit. Pro výpočet agregovaných hodnot pro každou skupinu na vybrané úrovni bylo nejprve nutné shlukovat autory, dokud nebudou rozděleni do skupin vybrané úrovně. Pro shlukování autorů jsem chtěl použít algoritmus pro aglomerativní hierarchické shlukování, kde podobnost by byla jasně dána rodičovskou skupinou shlukovaných skupin autorů. Kvůli již dané stromové struktuře, kde listové uzly nejsou vždy na stejné úrovni, jsem musel od algoritmu shlukování upustit a využil jsem opět rekurzivní preorder algoritmus pro průchod podstromu. Průchod podstromu se provádí již bez kořene, tím jsou procházeny jen uzly na nižší úrovni a autoři všech nižších úrovní jsou přiřazeni ke skupině vybrané úrovně.

Jedním z úkolů požadovaných funkcí 4 a 5 uvedených v kapitole 3 bylo umožnit zobrazení detailních informací autora po kliknutí na jeho jméno v tabulce. Pro zobrazení detailu autora jsem využil možnosti dialogového okna, které poskytuje *Bootstrap*. Detailní informace autora jsou do dialogového okna vždy získány z databáze a hodnoty jsou poté naplněny do ovládacích prvků *Label*. Pro získání těchto dat jsem zde využil technologii AJAX pro asynchronní volání. Uživatel si tak ani nevšimne, že se provedlo volání metod ze serveru, čímž je práce s aplikací uživatelsky příjemnější. Pro implementaci asynchronního volání jsem využil prvek UpdatePanel. Protože jsou identifikátory prvků se jmény autorů generovány dynamicky, nešlo tak jednoduše přidělit odkazům funkci spouštěče částečného *PostBacku*. Proto je tato funkce přidávána odkazů vždy po dokončení načtení stránky v metodě *Page_LoadCompleted*. Otevření dialogového okna se již provádí jednoduchou funkcí pomocí JavaScriptu 6.4.7.

Všechny tabulky s autory také umožňují seřazovat hodnoty dle libovolného atributu, což s sebou nese potřebu uchování seřazovacího výrazu a směru seřazení. Pro uchování těchto parametrů jsem si vytvořil vlastnosti, které je uchovávají ve *ViewState*.

Dalším z úkolů požadovaných funkcí 9 kapitoly 3 bylo zobrazit v administraci počty publikací po vepsání jednotlivých ID autora. Při vepsání ScopusId se získávají počty publikací ze Scopus Search API. V případě ReasearcherId a OrcId jsou počty publikací získávány z Web of Science. Počet publikací je uveden na stránce se seznamem autorů, ale v HTML dokumentu vráceném v HTTP odpovědi serveru tento počet nebyl zobrazen. Nejprve jsem si myslel, že je tento počet dodatečně dopočítán a zobrazen, ale po chvíli hledání v HTML dokumentu jsem nalezl skryté pole, ve kterém byl počet publikací uložen. Pro získání dat jsem opět využil asynchronního volání a po získání výsledku počet zobrazen uživateli pomocí prvku UpdatePanel.

Všechny administrační formuláře jsou také řádně validovány. V případě úprav hodnot přímo v tabulce jsou validační zprávy vypsány do prvku ValidationSummary a v případě formulářů jsou vepsány hned vedle pole pro vyplnění. Ošetřeny jsou také roky při aktualizaci a deaktivaci autora, aby nenastala situace, že by měl autor nastaven rok příchodu na univerzitu vyšší než je rok jeho odchodu z univerzity.

8 Webová aplikace

Tato kapitola již ukazuje výslednou funkčnost systému s ukázkami uživatelského rozhraní jednotlivých formulářů a popisuje také možnosti, které aplikace uživateli nabízí. Taktéž dále popisuje administrativní část aplikace. Do administrační části aplikace se dostane jen uživatel v roli administrátora, ten pak má přístup ke všem čtyřem administračním formulářům.

Finální verze webové aplikace je složena z devíti formulářů. Čtyři formuláře slouží pro administraci aplikace a 5 je určeno pro uživatele, ale formulář pro zobrazení informací o publikacích a autorech je popsán v kolegově práci [3], protože jej vytvářel on. Proto zde popisuji zbylé 4 formuláře a jejich funkci v rámci systému. Prvním je výchozí stránka aplikace, kde jsou zobrazeny jen některé informace o publikacích a citacích, další dva formuláře slouží jako detailní stránky pro autora a publikace, kde si může uživatel zobrazit citační ohlas dle jednotlivých let a poslední stránkou je formulář pro zobrazení statistik citačního ohlasu autorů a skupin. Čtyři formuláře určené pro administraci aplikace slouží k přidávání nových autorů, rozdělování autorů do skupin a také slouží ke správě automatických aktualizací dat nebo úpravě pracovních zařazení.

8.0.1 Stránka s přehledem

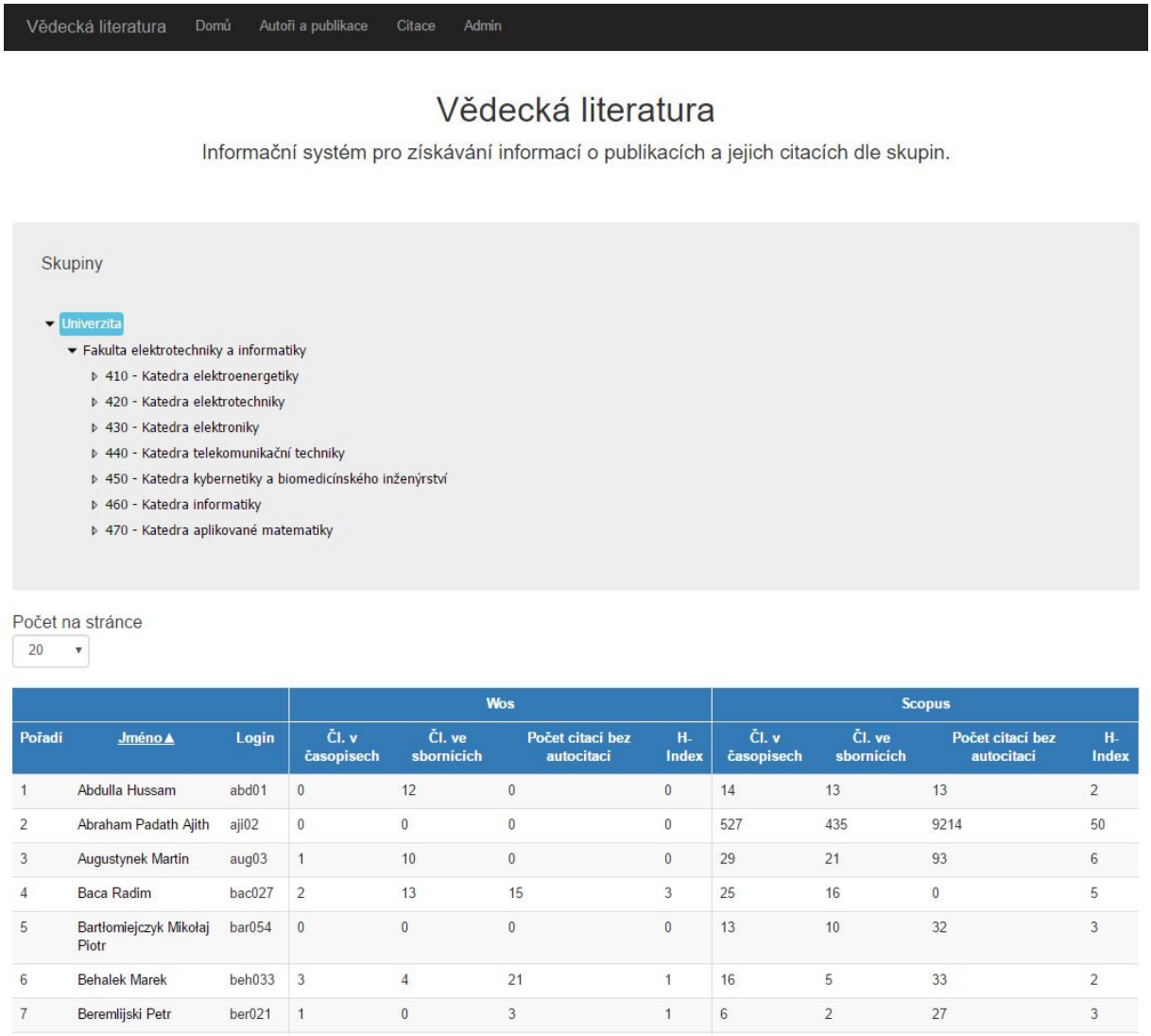
Tato stránka se zobrazí jako první po přístupu k aplikaci. Jedná se o formulář pro zobrazení základních informací o publikacích i citacích autorů. Seznam skupin ve stromové struktuře umožňuje uživateli výběrem skupiny filtrovat data v tabulce s autory. Tabulka je stránkovaná a uživatel si může vybrat, kolik autorů chce na stránce zobrazit. Data jsou do tabulky načtena vždy ihned po změně filtrované skupiny nebo po výběru počtu autorů k zobrazení. Formulář je vyobrazen na obrázku 12.

Po kliku na jméno autora v tabulce je uživateli zobrazeno dialogové okno s detailními informacemi autora viz obrázek 13. Stejné okno se zobrazí uživateli i po kliku na jméno v tabulce ve formuláři citací autorů a skupin autorů i ve formuláři s publikačními informacemi.

8.1 Stránka citací autorů a skupin autorů

Formulář pro skupiny autorů je z pohledu mého zadání hlavní stránka celé aplikace, umožňuje jednoduché zobrazení statistik citačního ohlasu autorů dle skupin a také umožňuje zobrazení statistik citačního ohlasu celých skupin autorů.

Filtrační část formuláře 14 umožňuje uživateli výběr skupiny pro zobrazení autorů a souhrnných statistik citací. Při výběru skupiny jsou uživateli zobrazeni autoři vybrané skupiny a všech jejich podskupin. V případě, že vybraná skupina obsahuje podskupiny, má uživatel možnost zvolit si úroveň podskupiny, do kterých mají být vypočítány souhrnné informace společně se souhrnem vybrané skupiny. Dále zde uživatel může filtrovat výsledné autory a tudíž i souhrnné informace podle pracovního zařazení autora nebo si zde může vybrat, jestli chce zahrnout kompletní citační ohlas publikací neaktuálních autorů nebo jen citace publikací autorů publi-



			Wos				Scopus			
Pořadí	Jméno▲	Login	Čl. v časopisech	Čl. ve sbornících	Počet citací bez autocitací	H-Index	Čl. v časopisech	Čl. ve sbornících	Počet citací bez autocitací	H-Index
1	Abdulla Hussam	abd01	0	12	0	0	14	13	13	2
2	Abraham Padath Ajith	aji02	0	0	0	0	527	435	9214	50
3	Augustynek Martin	aug03	1	10	0	0	29	21	93	6
4	Baca Radim	bac027	2	13	15	3	25	16	0	5
5	Bartłomiejczyk Mikołaj Piotr	bar054	0	0	0	0	13	10	32	3
6	Behalek Marek	beh033	3	4	21	1	16	5	33	2
7	Beremlijski Petr	ber021	1	0	3	1	6	2	27	3

Obrázek 12: Uživatelské rozhraní stránky s přehledem webové aplikace

kovaných v době úvazku na univerzitě. Všechny filtrační funkce jsou aplikovány až po stisku tlačítka *Zobrazit* včetně vybrané velikosti stránky tabulky s autory.

Souhrnná tabulka je rozdělena na celkové a průměrné hodnoty. Obě podkategorie zobrazují počty citací, průměrné hodnoty navíc zobrazují h-index. Všechny podkategorie jsou rozděleny na citace s vlastními a bez vlastních citací autora. V řádcích jsou zobrazeny výsledné počty hodnot autorů vybrané skupiny, případně i jejich podskupin, pro Scopus a Web of Science. V případě, že uživatel vybere pro zobrazení skupinu, která obsahuje podskupiny, pak mu aplikace zobrazí souhrnné statistiky v tabulce pro vybranou úroveň podskupin a současně zobrazí i souhrnné informace pro vybranou skupinu, což je vidět na obrázku 15.

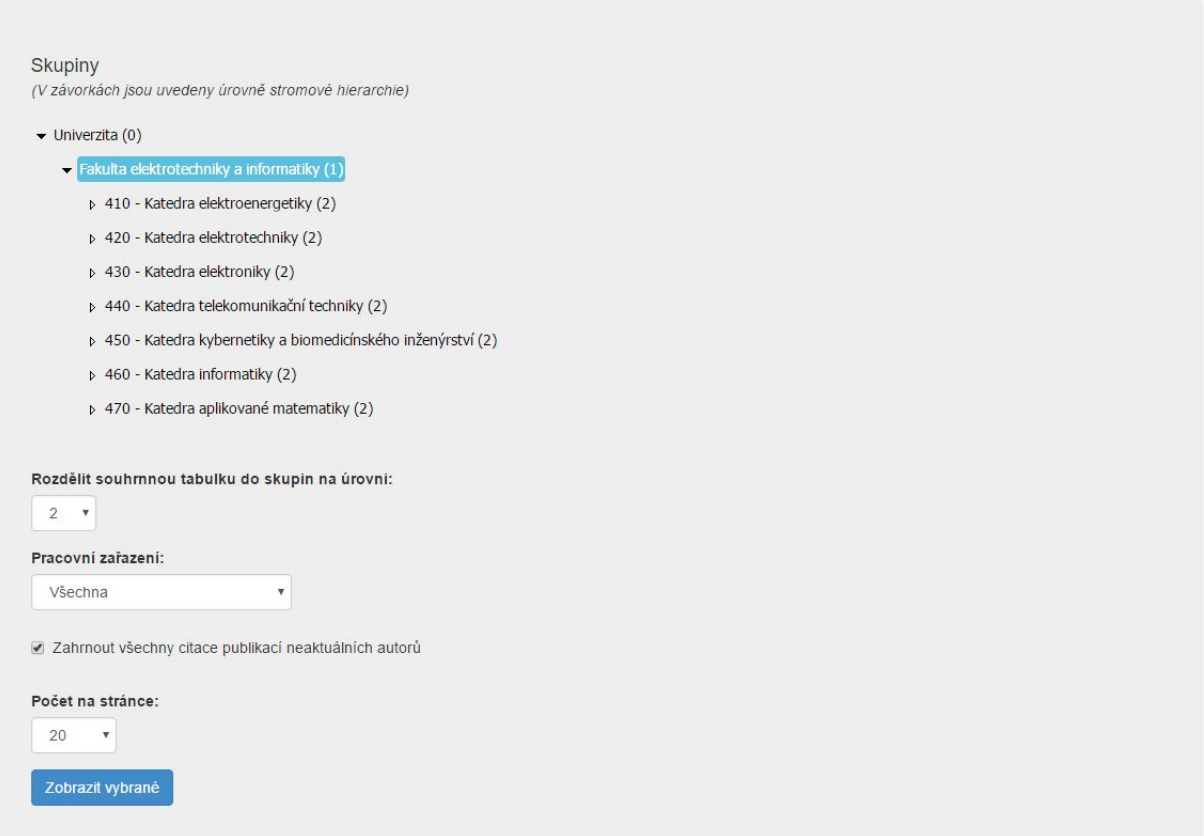
Hlavní rozdílem tabulky s autory oproti souhrnné tabulce je, že v řádcích se objevují autoři, zatímco hodnoty pro Scopus a Web of Science jsou vykresleny ve sloupcích. Hodnoty ukazatele

Obrázek 13: Uživatelské rozhraní dialogového okna s detailem autora

h-index jsou pro autory vypočítávány algoritmem popsaným v kapitole 5. Hodnoty v tabulce lze seřadit podle všech atributů, kromě pořadového čísla. Tabulka umožňuje přejít odkazem v posledním sloupci do Web of Science nebo Scopus s již vyhledanými publikacemi autora. Ukázka uživatelského rozhraní pro zobrazení autorů katedry elektroenergetiky je na obrázku 16

8.2 Formulář adminsitrace autorů

Tento formulář umožňuje administrátorovi přidávat, upravovat a nastavovat autory jako neaktuální. Všechna pole jsou povinná kromě OrcID a ResearcherID. Po vyplnění ScopusID ve formuláři je administrátorovi zobrazen počet publikací ve Scopus pro ujištění správnosti vyplněných hodnot. Stejně tak je zobrazen počet publikací ve Web of Science po vyplnění OrcID a ResearcherID. V další části formuláře je tabulka s autory a jejich vyplněnými informacemi. Autory v tabulce lze seřazovat dle libovolného atributu, filtrovat výběrem skupiny ze seznamu nad tabulkou nebo upravit počet autorů k zobrazení. Pro aktualizaci informací o autorovi musí administrátor vybrat autora v tabulce, aplikace vyplní autorovy hodnoty zpět do formuláře a umožní aktualizaci dat. Výběrem autora v tabulce aplikace umožní také nastavení autora jako neaktuálního, kde administrátor musí vybrat rok odchodu autora z univerzity. Celý formulář je vidět na obrázku 17 a 18.



Obrázek 14: Filtrační část formuláře skupin autorů

8.3 Formulář administrace skupin

Formulář umožňuje uživateli přidávat, upravovat a mazat skupiny. Jelikož jsou skupiny tvořeny ve stromové struktuře, musí administrátor při vytváření vybrat nadřazenou skupinu ze seznamu skupin. V případě, že se administrátor pokusí smazat skupinu, která je již přiřazena autorům, aplikace zobrazí v části pro smazání skupiny druhý seznam skupin pro výběr náhradní skupiny, která bude autorům přiřazena. Formulář pro administraci skupin je vidět na obrázku 19.

8.4 Formulář administrace pracovních zařazení

Posledním formulářem v administraci aplikace je administrace pracovních zařazení, která také umožňuje přidávání, úpravu a mazání pracovního zařazení. Aktualizace pracovního zařazení na rozdíl od předešlých formulářů se provádí přímo v tabulce pracovních zařazení, což je vidět na obrázku 20.

Souhrn

	Celkové hodnoty			Průměrné hodnoty					
	Počet citací			Počet citací			H-Index		
Zdroj	S autocitacemi	Bez autocitací autora	Bez autocitací všech autorů	S autocitacemi	Bez autocitací autora	Bez autocitací všech autorů	S autocitacemi	Bez autocitací autora	Bez autocitací všech autorů
Fakulta elektrotechniky a informatiky									
Scopus	29844	22389	18790	151,5	113,6	95,4	4,2	3,4	3
Wos	7664	5501	1168	38,9	27,9	5,9	2,4	2	0,4
410 - Katedra elektroenergetiky									
Scopus	874	520	447	41,6	24,8	21,3	2,8	2	2
Wos	569	357	6	27,1	17	0,3	2,4	1,9	0,1
420 - Katedra elektrotechniky									
Scopus	596	432	316	31,4	22,7	16,6	2,5	2,2	2,1
Wos	681	591	0	35,8	31,1	0	2	1,7	0
430 - Katedra elektroniky									
Scopus	805	737	666	73,2	67	60,5	4,5	4,4	4
Wos	321	300	0	29,2	27,3	0	2,7	2,6	0
440 - Katedra telekomunikační techniky									
Scopus	1675	805	553	69,8	33,5	23	3,5	2,4	2
Wos	449	169	77	18,7	7	3,2	1,7	1,2	0,4
450 - Katedra kybernetiky a biomedicínského inženýrství									
Scopus	3359	2272	1803	81,9	55,4	44	4	3,4	2,9
Wos	1759	1188	0	42,9	29	0	2,6	2,1	0
460 - Katedra informatiky									
Scopus	17762	14268	12464	355,2	285,4	249,3	5,5	4,3	3,9
Wos	2401	1763	69	48	35,3	1,4	2,7	2,2	0,2
470 - Katedra aplikované matematiky									
Scopus	4773	3355	2541	154	108,2	82	5	4,1	3,6
Wos	1484	1133	1016	47,9	36,5	32,8	2,5	2	1,9

Obrázek 15: Uživatelské rozhraní tabulky se statistikami skupin autorů

Souhrn

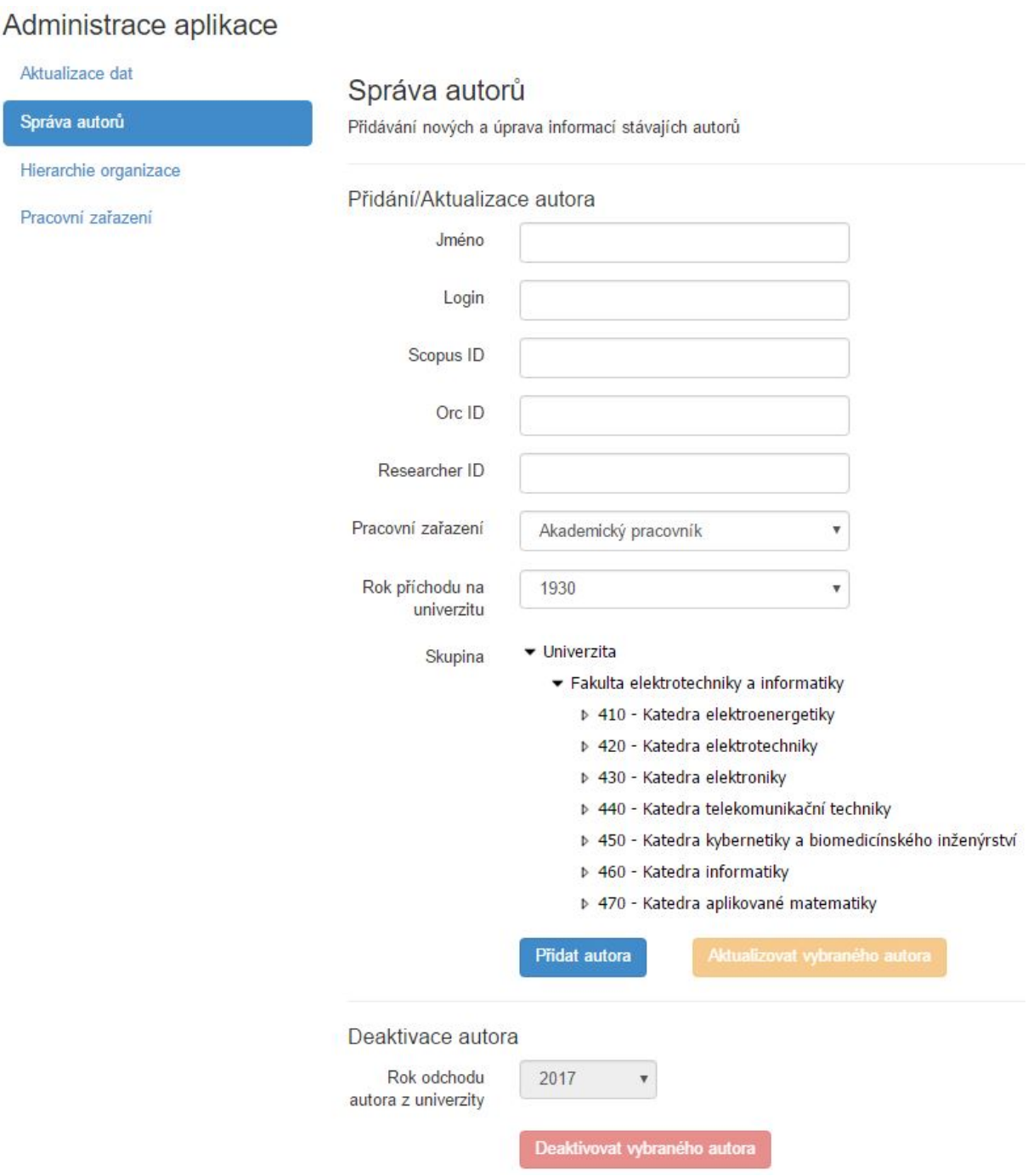
	Celkové hodnoty			Průměrné hodnoty					
	Počet citací			Počet citací			H-Index		
Zdroj	S autocitacemi	Bez autocitací autora	Bez autocitací všech autorů	S autocitacemi	Bez autocitací autora	Bez autocitací všech autorů	S autocitacemi	Bez autocitací autora	Bez autocitací všech autorů
Scopus	874	520	447	41,6	24,8	21,3	2,8	2	2
Wos	569	357	6	27,1	17	0,3	2,4	1,9	0,1

Autoři

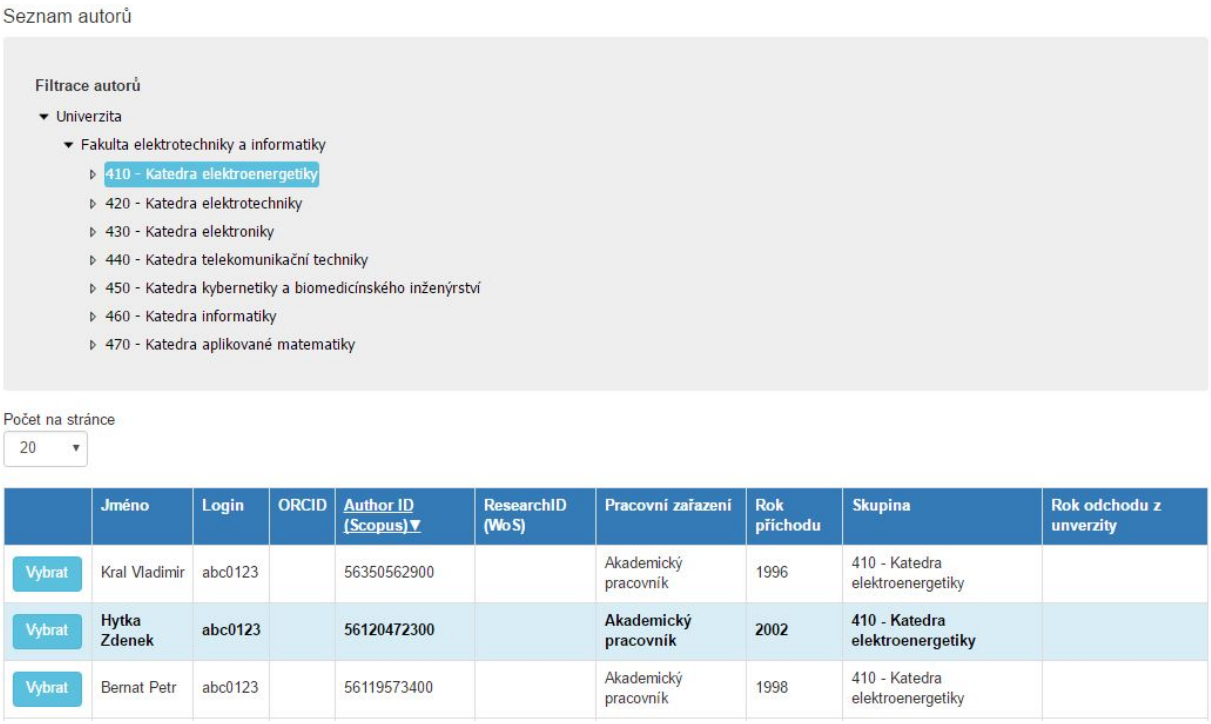
* Link - kliknutím na tlačítko přejít proběhne přesměrování na seznam publikací daného autora v publikační databázi

			S autocitacemi				Bez autocitací								Link*	
			Počet citací		H-Index		Počet citací				H-Index					
							Daného autora		Všech autorů		Daného autora		Všech autorů			
Pořadí	Jméno▲	Login	Scopus	Wos	Scopus	Wos	Scopus	Wos	Scopus	Wos	Scopus	Wos	Scopus	Wos	WoS	Scopus
1	Bernat Petr	abc0123	6	4	1	1	3	4	3	0	1	1	1	0	Přejít	Přejít
2	Cech Vaclav	abc0123	0	0	0	0	0	0	0	0	0	0	0	0	Přejít	Přejít
3	Gala Jan	abc0123	0	0	0	0	0	0	0	0	0	0	0	0	Přejít	Přejít
4	Gono Radomir	abc0123	95	44	5	3	35	28	56	0	3	3	4	0	Přejít	Přejít
5	Gurecky Jiri	abc0123	13	4	2	2	2	2	2	0	1	1	1	0	Přejít	Přejít
6	Hradilek Zdenek	abc0123	40	12	3	2	18	6	9	4	2	1	1	1	Přejít	Přejít
7	Hytka Zdenek	abc0123	2	2	1	1	2	2	2	0	1	1	1	0	Přejít	Přejít
8	Chmelik Karel	abc0123	1	3	1	1	1	3	1	0	1	1	1	0	Přejít	Přejít
9	Kacor Petr	abc0123	35	15	3	3	27	12	21	0	3	2	2	0	Přejít	Přejít

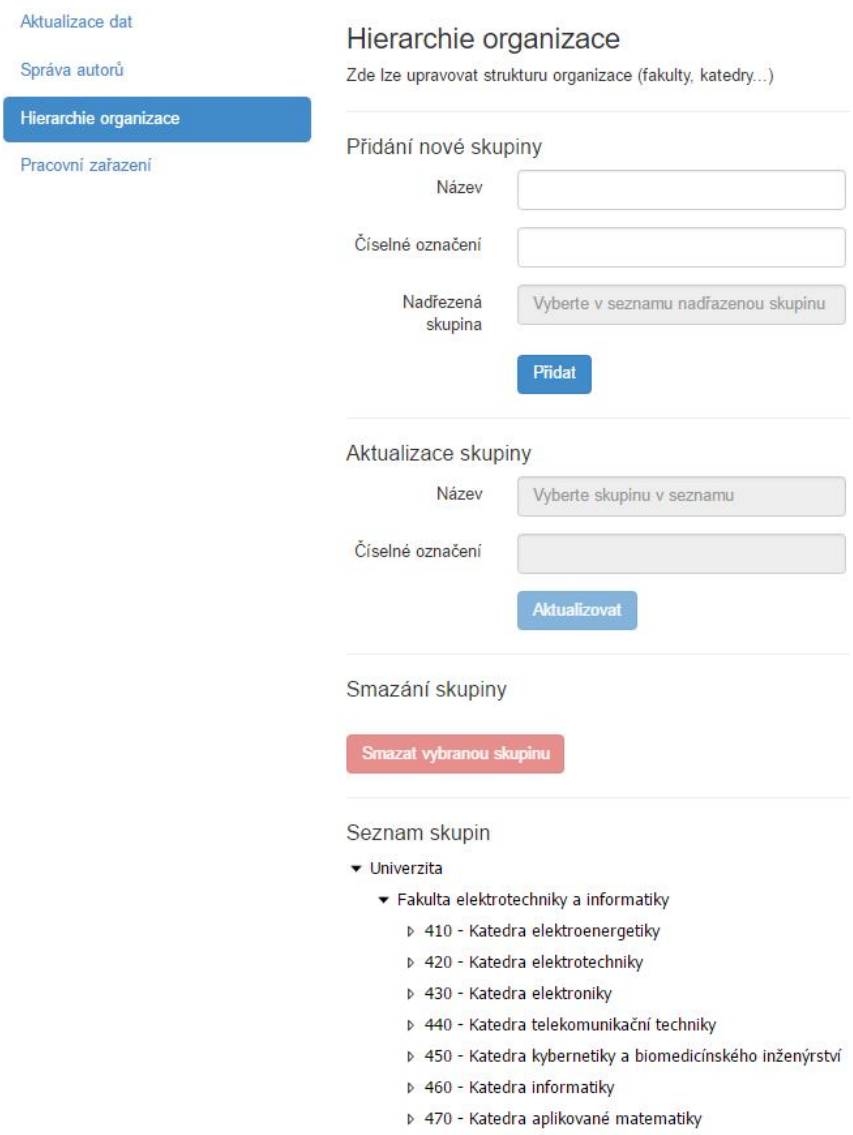
Obrázek 16: Uživatelské rozhraní stránky pro statistiky skupin autorů



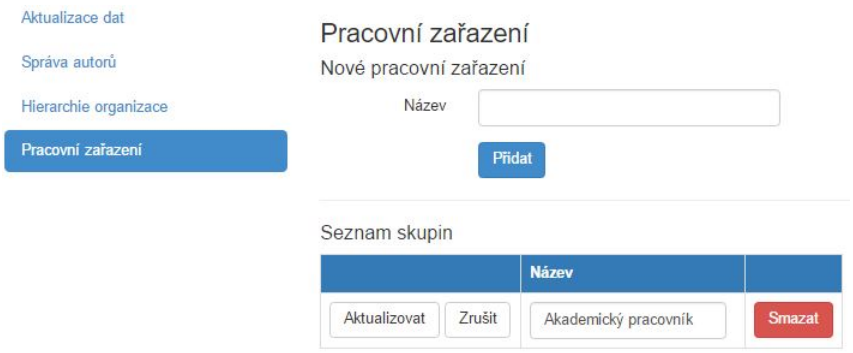
Obrázek 17: Uživatelské rozhraní administrace autorů



Obrázek 18: Uživatelské rozhraní seznamu autorů v administraci



Obrázek 19: Uživatelské rozhraní administrace skupin



Obrázek 20: Uživatelské rozhraní administrace pracovních zařazení

9 Měření časů získávání dat

V této kapitole je popsáno 6 měření časů. Prvním je měření časů optimalizace implementace, druhým měřením časů je získávání dat z Web of Science a Scopus API. Také jsem změřil získávání počtu publikací autora, jelikož na tyto výsledky administrátor čeká v reálném čase. Dále jsem naměřil časy výpočtu ukazatelů h-index pro data získaná z obou citačních databází a jako poslední jsou uvedeny naměřené časy získávání dat z vlastní databáze. Poslední tabulka slouží pouze pro porovnání časů získávání dat z API (resp. webu) a z vlastní databáze. Pro přesné měření časů jsem spouštěl aplikaci v release módu.

9.1 Porovnání časů způsobu uchování dat před a po optimalizaci

V kapitole 7 jsem již popsal optimalizaci uchovávání dat při jejich získávání z Web of Science. U obou implementací způsobu uchovávání dat jsem naměřil časy získávání dat pro dva autory. Zvolil jsem k porovnání autory doc. Ing. Miroslava Vozňáka, Ph.D., jehož jménu ve Web of Science odpovídá 90 publikací a Ing. Jana Rozhona, Ph.D., jemuž odpovídá 21 publikací. Tito autoři spolupracovali celkem na 21 publikacích, jejichž citující publikace se v prvním způsobu implementace procházely dvakrát. Čas před optimalizací implementace a po optimalizaci jsem porovnal v následující tabulce 3, je zde uveden i počet publikací, které bylo potřeba projít v obou verzích.

Tabulka 3: Porovnání před a po změně struktury uchovávání publikací

	Seznam	**Slovník**
Naměřený čas (mm:ss)	09:19	07:20
Počet prohlížených publikací	211	189

Z tabulky lze vyčíst, že jsem se rozhodl správně pro optimalizaci. Ušetřil jsem tak pouze na dvou autorech průchod 22 citujících publikací a rozdíl v naměřených časech je 119 sekund, což je 21% z času původní implementace se seznamem.

9.2 Časy získávání dat ze Scopus API a Web of Science a časy výpočtů ukazatelů

Jedním z požadavků vedoucího práce bylo vytvořit funkcionalitu pro automatické aktualizace všech dat ve vlastní databázi aplikace s využitím již existující implementace pro získávání dat. Tímto úkolem se zabývá kolega ve své práci [3]. Aby mohl kolega správně nastavit limit pro automatické aktualizace, bylo nezbytné změřit získávání citačního ohlasu z obou citačních databází. Protože jsem měl k dispozici seznam kateder Fakulty elektrotechniky a informatiky spolu s jejich autory, měřil jsem čas získávání dat na tomto vzorku.

Výsledné časy získávání dat z Web of Science jsou uvedeny v tabulce 4 a časy získávání dat ze Scopus API jsou uvedeny v tabulce 5

Tabulka 4: Časy získávání citačního ohlasu dle skupin z Web of Science

Katedra	**Počet autorů**	**Počet publikací**	**Počet citujících publikací**	**Naměřený čas (hh:mm:ss)**
410	19	626	569	06:50:30
420	18	287	681	04:53:44
430	11	123	321	00:39:09
440	23	282	439	00:42:19
450	38	686	1759	09:36:59
460	49	1108	2401	04:39:19
470	28	321	1484	00:38:43
Celkem	**186**	**3433**	**7654**	**28:01:43**

Tabulka 5: Časy získávání citačního ohlasu dle skupin ze Scopus API

Katedra	**Počet autorů**	**Počet publikací**	**Naměřený čas (mm:ss)**
410	19	540	03:11
420	18	285	01:48
430	11	158	01:08
440	23	406	03:13
450	38	853	05:32
460	49	2782	15:40
470	28	573	03:21
Celkem	**186**	**5597**	**33:53**

V tabulce 4 jde výrazné veliké rozdíly v získávání dat pro různé katedry. Jak jsem již uvedl v kapitole 7, při získávání dat jsou procházeny publikace autora, ale i jejich citující publikace. Důvodem tak značných rozdílů je optimalizovaný způsob uchování publikací při získávání dat. Ten umožňuje přeskočit průchod citujících publikací společných publikací autorů, o čemž jsem již psal v předchozí podkapitole.

Po získání dat je potřeba vypočítat také všechny 3 ukazatele h-index pro data z obou citačních databází. V tabulce 6 jsou uvedeny výsledné časy výpočtů všech tří ukazatelů najednou.

Tabulka 6: Časy výpočtů ukazatelů h-index

	Scopus	**Web of Science**
Naměřený čas (s)	0,084	0,088

V kapitole 7 jsem již psal, že v administrativní části aplikace při vytváření autora jsou získávány počty publikací ze Scopus API a Web of Science. Protože jsou tato data získávána a zobrazována administrátorovi aplikace v reálném čase, změřil i jejich časy. Data jsou získávána podle ID autora, v případě počtů publikací ve Web of Science jsou získávána podle *ResearcherId* a *OrcId*, proto jsem naměřil časy získávání dat pro oba identifikátory. Časy jsem měřil pro autora Ing. Jana Látala, Ph.D. z Katedry telekomunikační techniky. Naměřené časy a získaný počet publikací je uvedeny v tabulce 7.

Tabulka 7: Časy získávání počtu publikací autora

	Scopus	**Web of Science OrcId**	**Web of Science ResearcherId**
Naměřený čas (s)	0,753	4,345	4.359
Počet publikací	88	77	75

9.3 Časy získávání dat z databáze

Jak jsem již zmínil v kapitole 6, výsledná aplikace využívá pro uchování dat vlastní SQL databázi, která nabízí mnohonásobně vyšší propustnost než Scopus API nebo Web of Science. V tabulce 8 jsou uvedeny naměřené časy SQL dotazů, které jsou spouštěny před zobrazení dat ve formulářích aplikace.

Tabulka 8: Časy získávání citačního ohlasu dle skupin ze Scopus API

Získání dat pro	**Naměřený čas (s)**
Stránka s přehledem	0,049
Stránka s citacemi	0,067
Detail autora	0,003
Seznam autorů v administraci	0,007
Seznam skupin v administraci	0,001
Seznam pracovních zařazení v administraci	0,001

V poslední uvedené tabulce 9 jsou zobrazeny časy získávání dat ze Scopus API a Web of Science v porovnání s vlastní databází aplikace. Pro získání citačního ohlasu 19 autorů je rozdíl ve prospěch vlastní databáze, v porovnání se Scopus API, 4 řády a v případě Web of Science dokonce 6 řádů.

Tabulka 9: Porovnání naměřených časů získávání dat

Citační ohlas 19 autorů	**API (Web) (hh:mm:ss)**	**Vlastní DB (s)**
Scopus	00:03:11	0,032
Web of Science	06:50:30	

10 Závěr

Při výběru práce jsem o problematice vědeckých publikací a citací neměl povědomí. Většinu informací a pochopení problematiky se mi dostalo při analýze stávajícího řešení. Samotná analýza stávajícího řešení zabrala poměrně dost času, protože bylo nutné nastudovat a otestovat jednotlivá Scopus API a následně i Web of Science. Při vývoji výsledného systému nastaly mnohé komplikace, které jsem musel vyřešit. Při implementaci přístupového objektu ke Scopus API zabrala hodně času jen komunikace s podporou *Elsevier*. Například vyřízení povolení Citations Overview API trvalo 2 a půl měsíce. Dalším důvodem ke kontaktování podpory byly špatné hodnoty, které vracelo Author Retrieval API, což jsem vyřešil použitím Scopus Search API a Citations Overview API s výsledným výpočtem chybějících ukazatelů.

Původním návrhem získávání dat z Web of Science bylo získávat data z jeho API, což nám nebylo umožněno kvůli omezení předplatného univerzity. Více o problémech s API je uvedeno v [3]. Výsledným řešením bylo získávání dat přímo z webu Web of Science, kde jsou k dispozici i citační informace. Web of Science již neposkytuje většinu požadovaných ukazatelů a bylo nezbytné pro výpočet ukazatelů získávat mnoho dalších informací, jako jsou citující publikace a jejich autoři.

Mnoho času jsem také zanechal při implementaci získávání dat z Web of Science pomocí komponenty WebBrowser a dalších nástrojů pro tvorbu webových prohlížečů. Poměrně dlouhou dobu jsem s využitím komponenty WebBrowser počítal, než jsem zjistil, že komponentu použít nemohu. Poté jsem analyzoval a testoval implementace jiných nástrojů, které by umožnily podobnou funkcionalitu a nakonec po dohodě s vedoucím práce jsem využil tříd frameworku .NET pro tvorbu HTTP požadavků, pomocí kterých získávání dat funguje bez problémů.

Hlavní nevýhodou získávání dat přímo z webu oproti získávání dat z API je doba potřebná pro získání všech dat. Při implementaci jsem tak strávil velkou část času jen testováním vlastní implementace, hlavně v případě, kdy jsem měl v implementaci chybu, která se projevila až po několika hodinách získávání dat. Představu o době strávené testováním a měřením výsledných časů si lze udělat již z tabulek předchozí kapitoly 9.

Z naměřených časů v kapitole 9 je vidět, že získávání citací trvá necelých 33 minut a 53 sekund pro Scopus a 28 hodin pro Web of Science. Kvůli takto dlouhým časům získávání dat nelze nechat aktualizaci na administrátorovi a data musí být ve vlastní databázi aktualizována automaticky v pravidelných intervalech, o čemž je více v [3].

Výsledek získaných dat a následně vypočítaných agregačními funkcemi je viditelný v následujícím obrázku 21. Zde je zobrazena tabulka s citacemi prvních 10 autorů katedry elektroenergetiky seřazených dle počtu citací ve Scopus vzestupně.

Při pohledu na implementaci by se určitě našly místa pro zlepšení, ale nad implementací jsem uvažoval a volil jsem algoritmy pro získávání dat s co nejvyšší efektivitou a úsporou času, čehož je důkazem i optimalizovaný způsob uchování dat v paměti popsaný v kapitole 7 a naměřený v kapitole 9.

			S autocitacemi				Bez autocitací							
			Počet citací		H-index		Počet citací				H-index			
							Daného autora		Všech autorů		Daného autora		Všech autorů	
Pořadí	Jméno	Login	Scopus▼	Wos	Scopus	Wos	Scopus	Wos	Scopus	Wos	Scopus	Wos	Scopus	Wos
1	Misak Stanislav	abc0123	204	108	9	6	129	23	106	0	6	3	6	0
2	Prokop Lukas	abc0123	167	83	9	6	112	61	88	1	6	4	6	1
3	Gono Radomir	abc0123	95	44	5	3	35	28	56	0	3	3	4	0
4	Rusek Stanislav	abc0123	86	46	5	3	25	31	46	0	2	3	4	0
5	Sokansky Karel	abc0123	71	39	4	3	56	31	30	0	4	3	3	0
6	Hradilek Zdenek	abc0123	40	12	3	2	18	6	9	4	2	1	1	1
7	Moldrik Petr	abc0123	39	9	3	2	16	5	11	1	2	1	2	1
8	Krejci Petr	abc0123	38	25	3	3	32	21	22	0	3	3	3	0
9	Kacor Petr	abc0123	35	15	3	3	27	12	21	0	3	2	2	0
10	Kral Vladimir	abc0123	33	116	3	6	23	87	25	0	2	5	2	0

Obrázek 21: Tabulka s autory a jejich citačními informacemi

Tato diplomová práce je poměrně dost spjatá s kolegovou diplomovou prací [3], protože výslednou aplikaci jsme tvořili oba. Odpovědi na některé otázky, které mohou vyvstat z této práce, jsou k nalezení v kolegově práci a naopak. Ale velikost průniku obou prací je minimální.

Vlastním přínosem výsledné práce je aplikace samotná, protože umožňuje funkčnosti, které nejsou umožněny v citačních databázích. Konkrétně se jedná o zobrazení citačních informací pro skupiny autorů jakékoliv úrovně struktury skupin, jejíž obsah tvoří administrátor. Také zobrazení citačního ohlasu autorů s možností filtrací dle skupin nebo pracovního zařazení.

Dalším přínosem práce je zobrazení ukazatelů, které nejsou v citačních databázích vůbec poskytovány, jako je pro Scopus počet citací bez autocitací všech autorů nebo h-index bez autocitací všech autorů. Pro Web of Science jich je ještě více, například h-index bez autocitací autora, h-index bez autocitací všech autorů, počet citací publikace bez autocitací autora a další.

Výsledný systém [3] tak hodnotím jako zdařilý nejen kvůli jeho přidané hodnotě, ale také kvůli aplikaci teoretických i praktických znalostí úrovně navazujícího studia jak v návrhu architektury systému, tak i způsobu jeho implementace.

Na výsledný systém lze navázat v případě jakékoliv další práce, která potřebuje využít stejná data jako jsou získávána přístupovými objekty, protože díky jejich návrhu jsou oddělitelná od projektu a znovu použitelná v jiném projektu.

Jiným způsobem navázání na mou práci je rozšíření systému o další citační databáze s nutností úpravy webové aplikace pro zobrazení těchto dat, ale bez nutnosti úprav již existujících přístupových objektů. Výsledný systém lze také rozšířit o uchování citací publikací v čase s možností grafického zobrazení meziročního nárůstu citačního ohlasu skupin autorů.

[3] Dopstupný na URL adrese `http://db.cs.vsb.cz/lit0016/`

Literatura

[1] SCOPUS - DOCUMENT SEARCH: *Scopus. [online]. [cit. 2017-02-23].* Dostupné z: `https://www.scopus.com/`

[2] WEB OF SCIENCE: *Web of Science Core Collection. [online]. [cit. 2017-02-23].* Dostupné z: `https://apps.webofknowledge.com/`

[3] KUCHARCZYK, Tomáš. *Systém pro získávání informací o vědeckých publikacích skupin autorů*. Ostrava, 2017. Diplomová práce. VŠB–Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky. Vedoucí diplomové práce doc. Ing. Michal Krátký, Ph.D..

[4] TKAČÍKOVÁ, Daniela. *JAK zpracovávat bibliografické citace podle normy ČSN ISO 690* [online]. Ostrava: VŠB-TU Ostrava, Ústřední knihovna, aktualizováno 2014-09-02 [cit. 2017-02-26].
Dostupné z: `http://knihovna.vsb.cz/kurzy/citace/`

[5] TKAČÍKOVÁ, Daniela. *Co je to h-index a jak ho zjistit v bázích dat Web of Science a Scopus* [online]. Ostrava: VŠB-TU Ostrava, Ústřední knihovna, aktualizováno 2017-01-03 [cit. 2017-02-26].
Dostupné z: `http://knihovna.vsb.cz/sluzby/h-index.htm`

[6] ELSEVIER: *Content - Scopus - Solutions. [online]. [cit. 2017-02-26].* Dostupné z: `https://www.elsevier.com/solutions/scopus/content`

[7] SCOPUS AUTHOR IDENTIFIER: *Scopus Author Identifier. [online]. [cit. 2017-04-14].* Dostupné z: `http://help.elsevier.com/app/answers/detail/a_id/2845/p/8150/c/7956`

[8] TKAČÍKOVÁ, Daniela. *Problémy s identifikací autorů* [online]. Ostrava: VŠB-TU Ostrava, Ústřední knihovna, aktualizováno 2017-04-24 [cit. 2017-04-26].
Dostupné z: `http://knihovna.vsb.cz/kurzy/uvod/12.html`

[9] CLARIVATE ANALYTICS: *Clarivate Analytics - Web Of Science. [online]. [cit. 2017-02-26].* Dostupné z: `http://clarivate.com/scientific-and-academic-research/research-discovery/web-of-science/`

[10] TKAČÍKOVÁ, Daniela. *Web of Science* [online]. Ostrava: VŠB-TU Ostrava, Ústřední knihovna, aktualizováno 2017-02-02 [cit. 2017-02-26].
Dostupné z: `http://knihovna.vsb.cz/sluzby/wos.htm`

[11] RESEARCHERID: *ResearcherId - Clarivate Analytics. [online]. [cit. 2017-04-20].* Dostupné z: `http://wokinfo.com/researcherid/`

[12] ORCID: *ORCID - Connecting Research and Researcher. [online]. [cit. 2017-04-20].* Dostupné z: `https://orcid.org/`

[13] ORCID: *ORCID - Statistics. [online]. Aktualizováno 2017-04-21 [cit. 2017-04-26].* Dostupné z: `https://orcid.org/statistics`

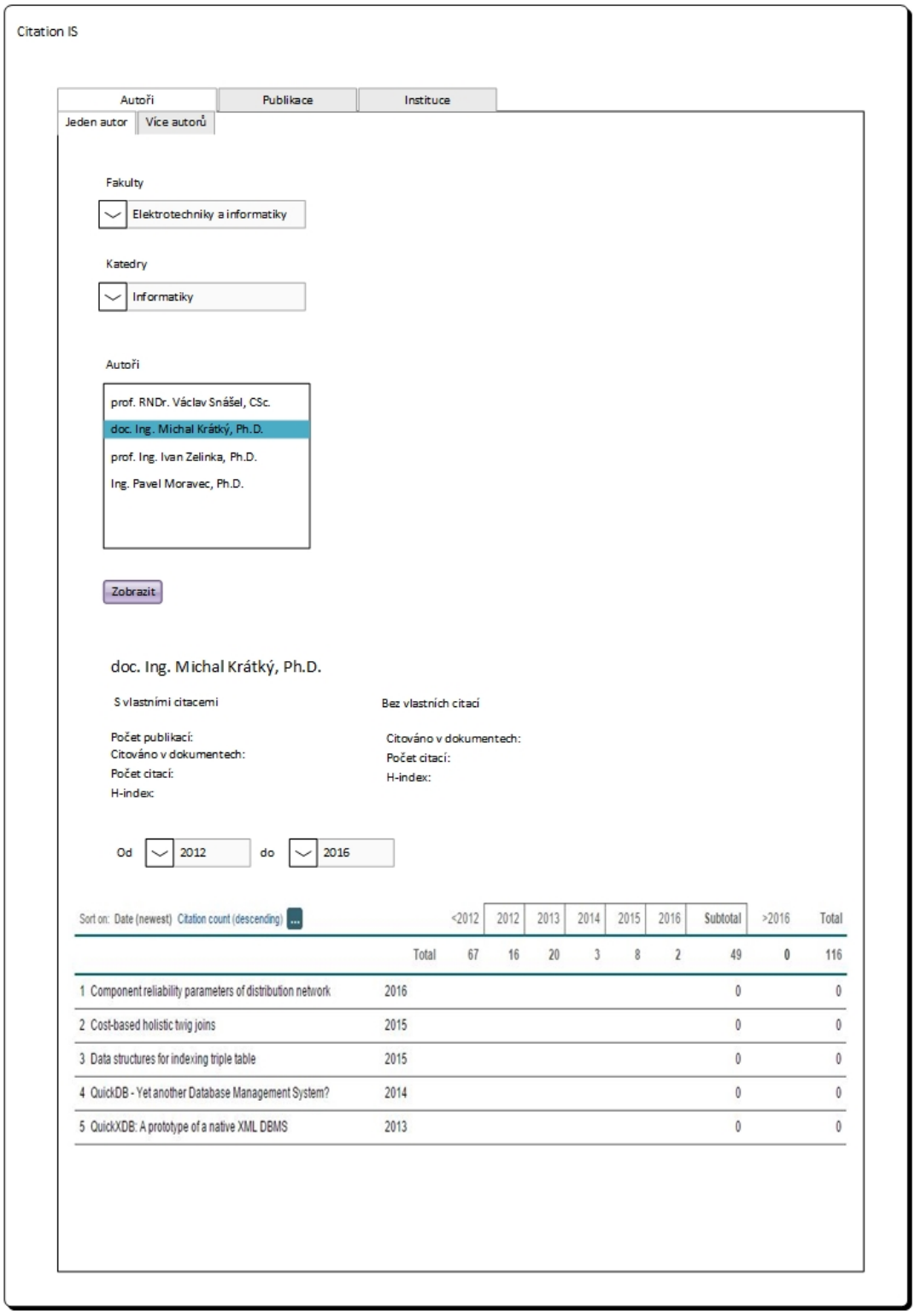
[14] ELSEVIER DEVELOPER PORTAL: *Elsevier Developer Portal. [online]. [cit. 2017-02-23].* Dostupné z: `https://dev.elsevier.com/`

[15] MICROSOFT DEVELOPER NETWORK: *Web and .NET Development. [online]. [cit. 2017-02-23].* Dostupné z: `http://msdn.microsoft.com/library`

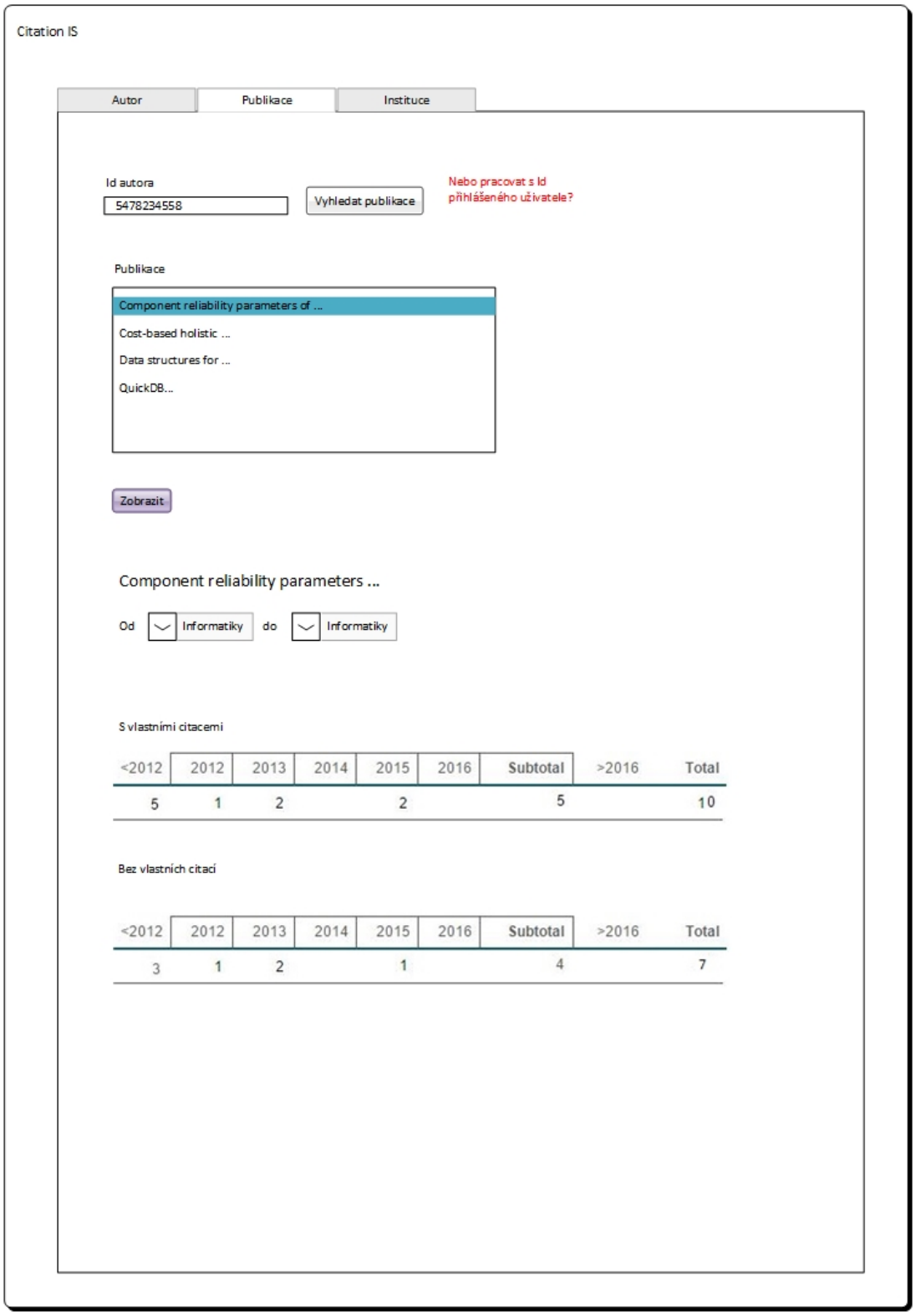
[16] WORLD WIDE WEB CONSORTIUM: *W3C. [online]. [cit. 2017-02-23].* Dostupné z: `https://www.w3.org/`

[17] HTML AGILITY PACK: *Html Agility Pack - Home. [online]. [cit. 2017-03-23].* Dostupné z: `https://htmlagilitypack.codeplex.com/`

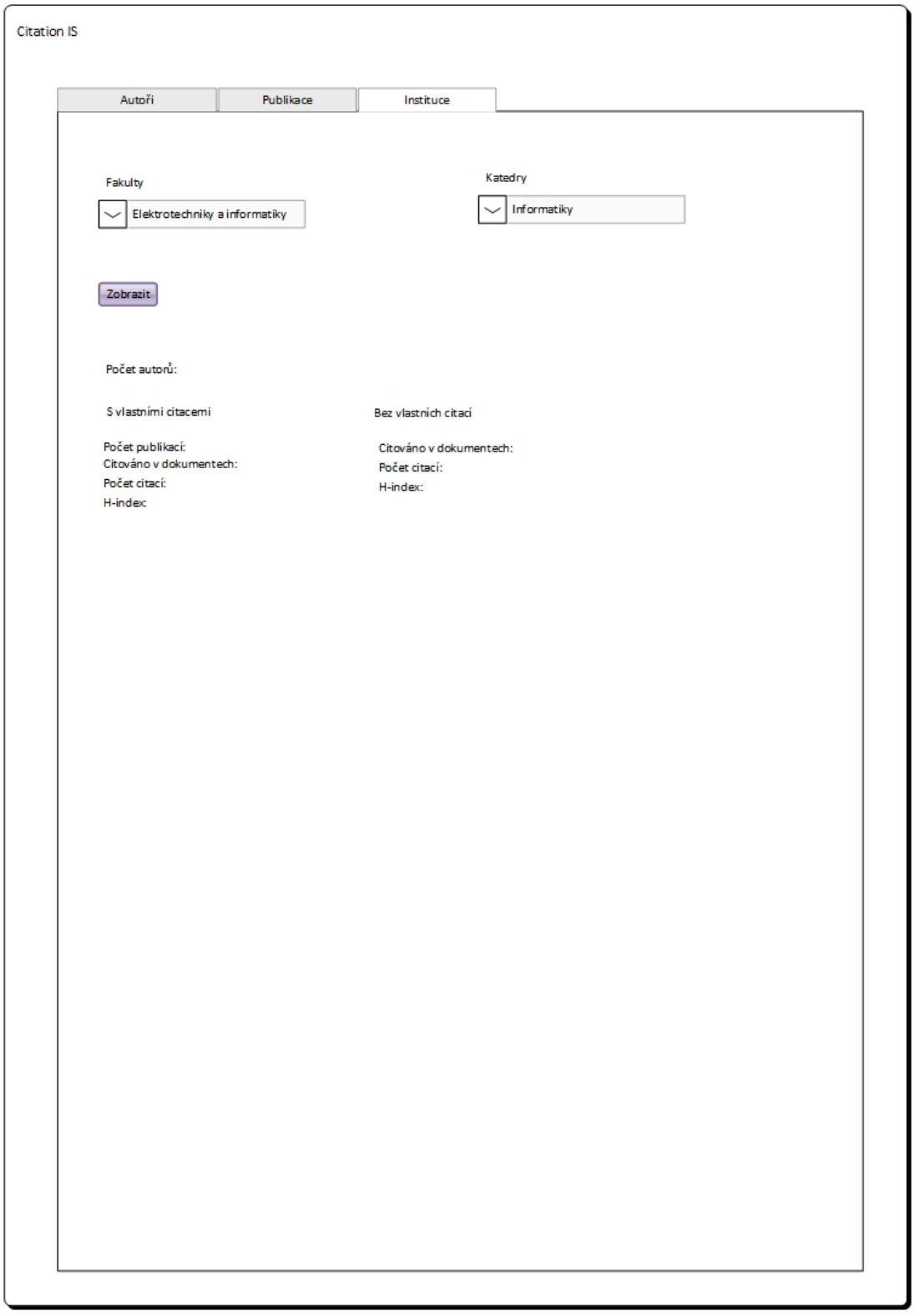
A Návrh webových formulářů



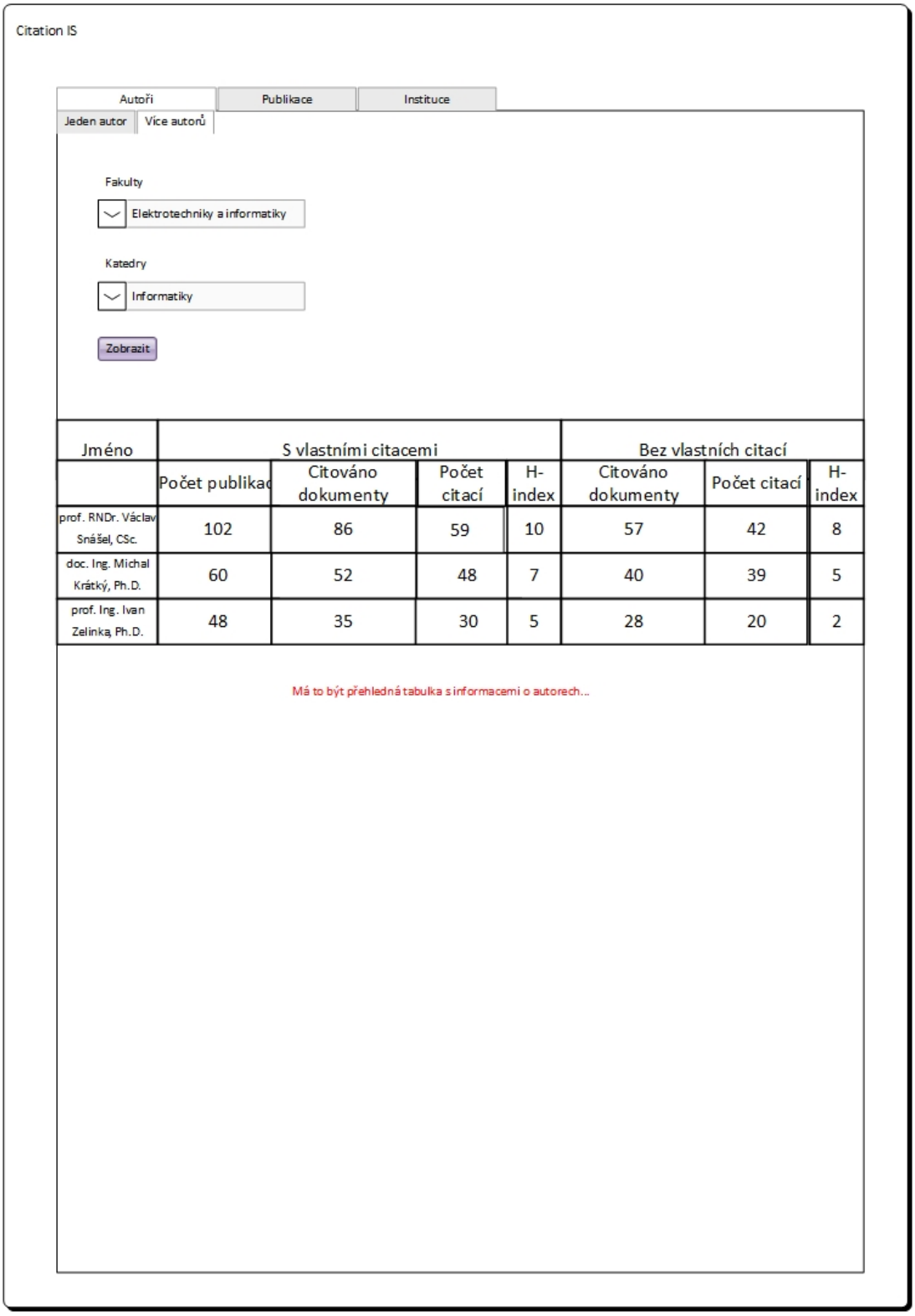
Obrázek 22: Návrh formuláře detailu autora



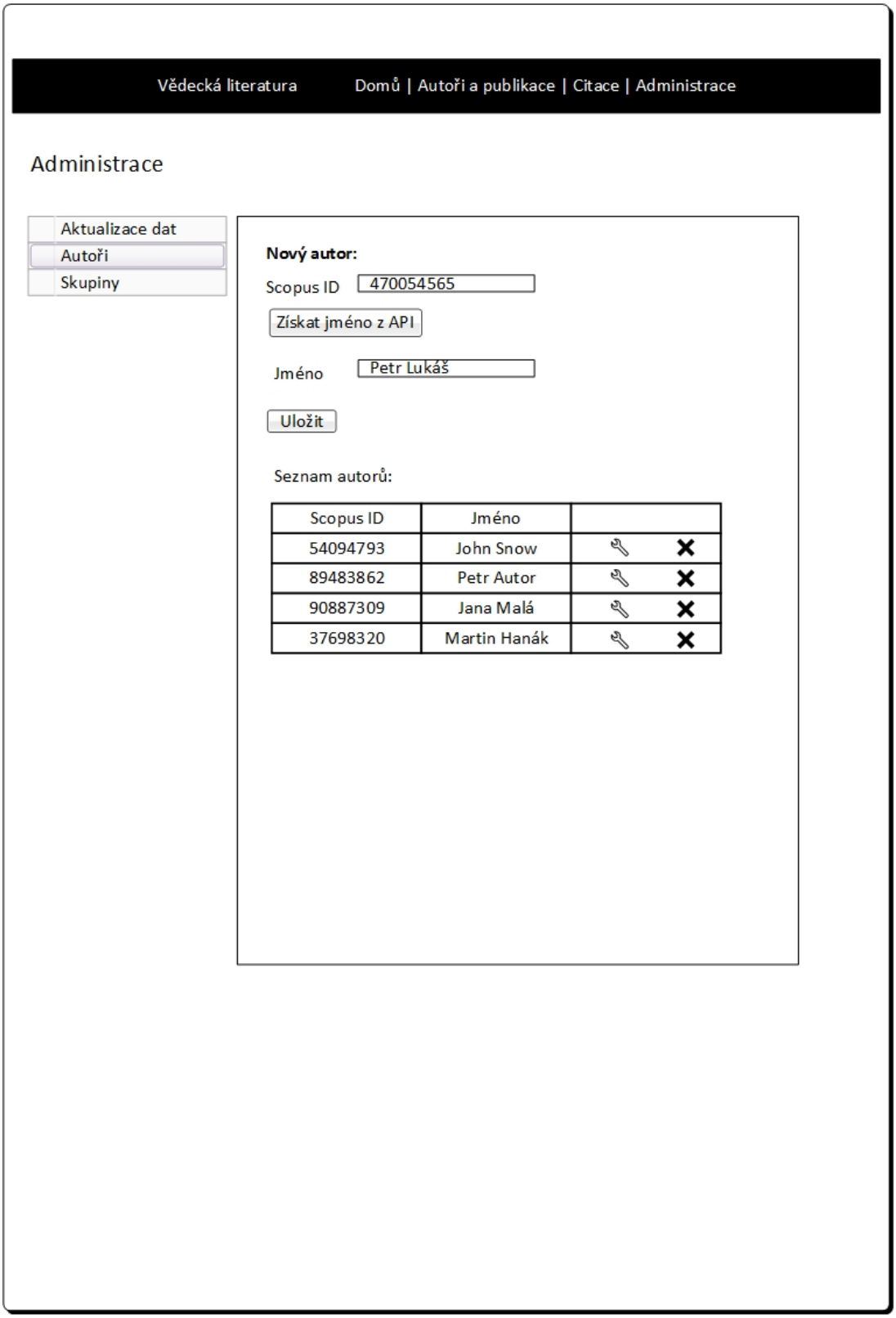
Obrázek 23: Návrh formuláře detilu publikace



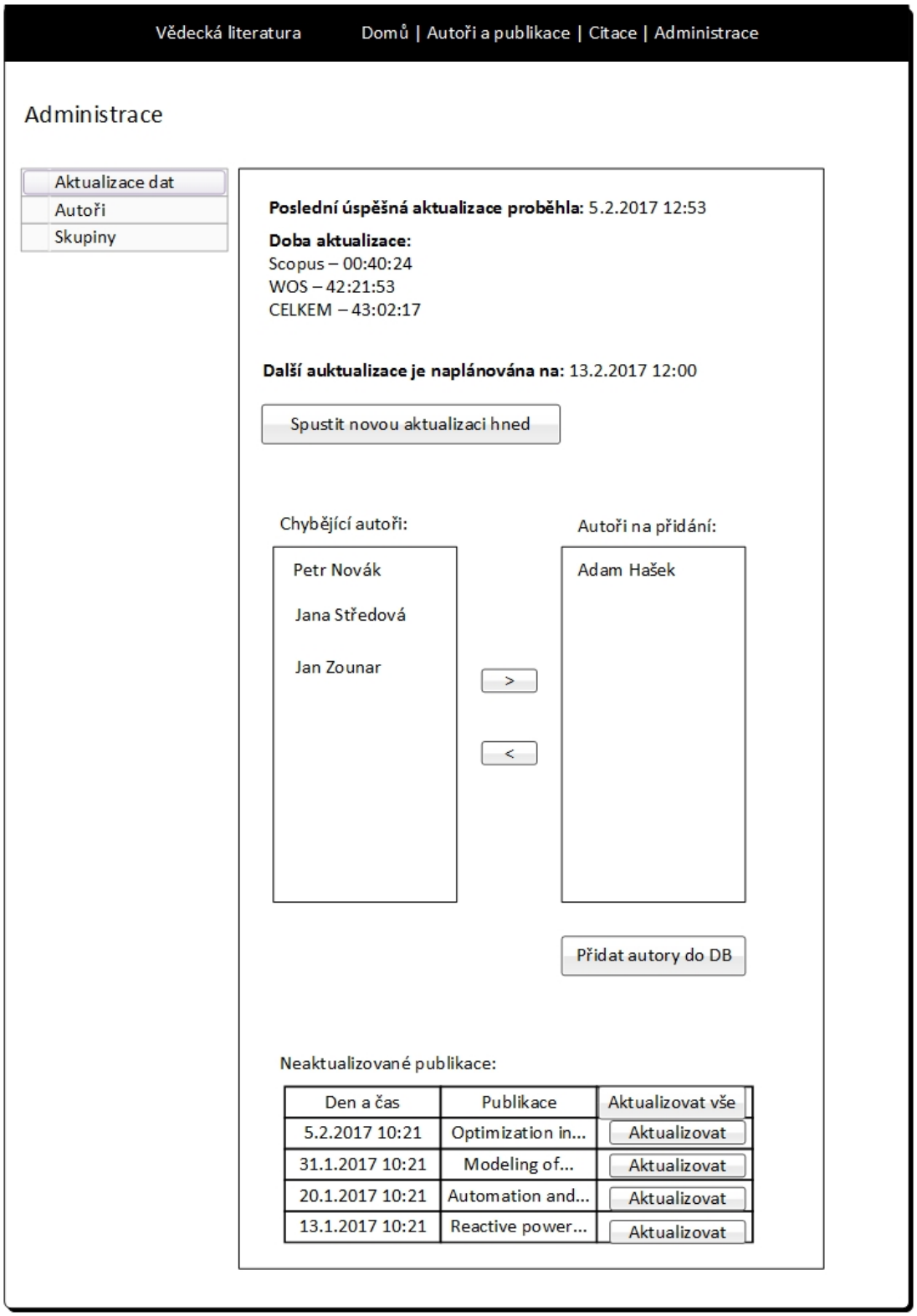
Obrázek 24: Návrh formuláře pro instituci



Obrázek 25: Návr formuláře pro skupiny autorů



Obrázek 26: Návr formuláře pro vytváření autorů v administrativní části



Obrázek 27: Návr formuláře pro aktualizaci dat v administrativní části

B XML dokument navrácený z Citations Overview API

```
<?xml version="1.0" encoding="UTF-8"?>
<abstract-citations-response xmlns:xocs="http://www.elsevier.com/xml/xocs/dtd"
    xmlns:ns2="http://webservices.elsevier.com/schemas/metadata/cto/types/v4"
    xmlns:ctom="http://www.elsevier.com/xml/ctom/dtd" xmlns:cto="http://www.
    elsevier.com/xml/cto/dtd" xmlns:prism="http://prismstandard.org/namespaces/
    basic/2.0/" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <h-index>1</h-index>
  <identifier-legend>
    <identifier>
      <dc:identifier>SCOPUS_ID:80055033085</dc:identifier>
      <prism:doi>10.1016/j.ins.2011.06.019</prism:doi>
      <pii>S0020025511003033</pii>
      <scopus_id>80055033085</scopus_id>
    </identifier>
  </identifier-legend>
  <citeInfoMatrix>
    <citeInfoMatrixXML>
      <citationMatrix xmlns="http://www.elsevier.com/xml/ctom/dtd">
        <citeInfo xmlns="">
          <dc:identifier>SCOPUS_ID:80055033085</dc:identifier>
          <prism:url>http://api.elsevier.com/content/abstract/
              SCOPUS_ID:80055033085</prism:url>
          <dc:title>Fast decoding algorithms for variable-lengths codes</
              dc:title>
          <author>
            <initials>J.</initials>
            <index-name>Walder J.</index-name>
            <surname>Walder</surname>
            <authid>36452537600</authid>
            <author-url>http://api.elsevier.com/content/author/
                AUTHOR_ID:36452537600</author-url>
          </author>
          <author>
            <initials>M.</initials>
            <index-name>Kratky M.</index-name>
            <surname> Krtk </surname>
            <authid>6701917792</authid>
```

```
  <author-url>http://api.elsevier.com/content/author/
      AUTHOR_ID:6701917792</author-url>
</author>
<author>
  <initials>R.</initials>
  <index-name>Baca R.</index-name>
  <surname>Baa</surname>
  <authid>23388475600</authid>
  <author-url>http://api.elsevier.com/content/author/
      AUTHOR_ID:23388475600</author-url>
</author>
<author>
  <initials>J.</initials>
  <index-name>Platos J.</index-name>
  <surname>Plato</surname>
  <authid>23397962200</authid>
  <author-url>http://api.elsevier.com/content/author/
      AUTHOR_ID:23397962200</author-url>
</author>
<author>
  <initials>V.</initials>
  <index-name>Snasel V.</index-name>
  <surname> Snel </surname>
  <authid>6602312912</authid>
  <author-url>http://api.elsevier.com/content/author/
      AUTHOR_ID:6602312912</author-url>
</author>
<citationType code="ar">Article</citationType>
<sort-year>2012</sort-year>
<prism:publicationName>Information Sciences</prism:publicationName>
<prism:volume>183</prism:volume>
<prism:issueIdentifier>1</prism:issueIdentifier>
<prism:startingPage>66</prism:startingPage>
<prism:endingPage>91</prism:endingPage>
<prism:issn>0020-0255</prism:issn>
<pcc>0</pcc>
<cc>7</cc>
<lcc>3</lcc>
<rangeCount>7</rangeCount>
```

```
          <rowTotal>10</rowTotal>
        </citeInfo>
      </citationMatrix>
    </citeInfoMatrixXML>
  </citeInfoMatrix>
  <citeColumnTotalXML>
    <citeCountHeader>
      <prevColumnHeading>previous</prevColumnHeading>
      <columnHeading>2013</columnHeading>
      <laterColumnHeading>later</laterColumnHeading>
      <prevColumnTotal>0</prevColumnTotal>
      <columnTotal>7</columnTotal>
      <laterColumnTotal>3</laterColumnTotal>
      <rangeColumnTotal>7</rangeColumnTotal>
      <grandTotal>10</grandTotal>
    </citeCountHeader>
  </citeColumnTotalXML>
</abstract-citations-response>
```

Výpis 10: Ukázka XML dokumentu

C SQL dotaz pro výběr všech autorů fakulty z databáze

```
SELECT g.Id, g.Code, g.Name, g.Level, a.Id, a.Name, a.[Login] , a.StartYear, a.
    ScopusHIndex, a.ScopusHIndexSelfExcluded, a.ScopusHIndexAllExcluded, a.
    WosHIndex, a.WosHIndexSelfExcluded, a.WosHIndexAllExcluded, q1.
    SumScopusCitationsCount, q1.SumScopusCitationsCountSelfExcluded, q1.
    SumScopusCitationsCountAllAuthorsfExcluded, q2.SumWosCitationsCount, q2.
    SumWosCitationsCountSelfExcluded, q2.
    SumWosCitationsCountAllAuthorsfExcluded
FROM [Author] a
JOIN [Group] g ON a.GroupId = g.Id
LEFT JOIN
   (SELECT a.Id, SUM(p.CitationsCount) AS SumScopusCitationsCount, SUM(ap.
      CitationsCountSelfExcluded) AS SumScopusCitationsCountSelfExcluded, SUM(p
      .CitationsCountAllAuthorsExcluded) AS
      SumScopusCitationsCountAllAuthorsfExcluded
   FROM Author a
   JOIN [AuthorPublications] ap ON a.Id = ap.AuthorId
   LEFT JOIN [Publications] p ON ap.PublicationId = p.Id
   WHERE p.ScopusId IS NOT NULL AND p.Published BETWEEN a.StartYear AND ISNULL(
      a.EndYear, YEAR(GetDATE()))
   GROUP BY a.Id) q1
   ON a.Id = q1.Id
LEFT JOIN
   (SELECT a.Id, SUM(p.CitationsCount) AS SumWosCitationsCount, SUM(ap.
      CitationsCountSelfExcluded) AS SumWosCitationsCountSelfExcluded, SUM(p.
      CitationsCountAllAuthorsExcluded) AS
      SumWosCitationsCountAllAuthorsfExcluded
   FROM Author a
   JOIN [AuthorPublications] ap ON a.Id = ap.AuthorId
   LEFT JOIN [Publications] p ON ap.PublicationId = p.Id
   WHERE p.WosId IS NOT NULL AND p.Published BETWEEN a.StartYear AND ISNULL(a.
      EndYear, YEAR(GetDATE()))
   GROUP BY a.Id) q2
   ON a.Id = q2.Id
WHERE 1=1 AND (g.Id = @GroupId OR g.Id = @GroupId1 OR g.Id = @GroupId2 OR g.Id
    = @GroupId3 OR g.Id = @GroupId4 OR g.Id = @GroupId5 OR g.Id = @GroupId6)
```

Výpis 11: SQL dotaz pro získání všech autorů fakulty

D Příloha na CD/DVD

- ScientificLiteratureSolution - složka obsahující projekty

  /Common

  /DataAccess

  /Entities

  /packages

  /ScientificLiteratureWeb

  /Scopus

  /WebOfKnowledge

  /WebOfKnowledgeBrowser

- SQL_Scripts - složka obsahující SQL skripty s databází